

Copyright

by

Xiaoyan Si

2015

**The Dissertation Committee for Xiaoyan Si Certifies that this is the approved  
version of the following dissertation:**

**Models and Methods for Operational Planning  
in Freight Railroads**

**Committee:**

---

Jonathan F. Bard, Supervisor

---

Anantaram Balakrishnan, Co-Supervisor

---

Ned Dimitrov

---

John J. Hasenbein

---

David P. Morton

**Models and Methods for Operational Planning  
in Freight Railroads**

**by**

**Xiaoyan Si, B.E.; M.S.E.**

**Dissertation**

Presented to the Faculty of the Graduate School of  
The University of Texas at Austin  
in Partial Fulfillment  
of the Requirements  
for the Degree of

**DOCTOR OF PHILOSOPHY**

**The University of Texas at Austin**

**May 2015**

## **Dedication**

To my husband.

## **Acknowledgements**

First and foremost, I must thank my co-supervisor, Dr. Anant Balakrishnan, for his tireless advising throughout the past five years. Without him, this dissertation would not have been possible. Dr. Balakrishnan is passionate and dedicated to his students' research, and my discussions with him were always illuminating. I am also thankful to Dr. Jonathan Bard for serving as my supervisor, and to Dr. John Hasenbein, Dr. Ned Dimitrov, and Dr. David Morton for taking the time to serve on my committee. I appreciate their support and enjoyed my conversations with them.

I would like to thank Dr. Pooja Dewan for providing me the opportunity to learn about the railroad industry and exposing me to many of its challenging and interesting problems for my research. I am also grateful to Dr. April Kuo for being my mentor and providing many valuable suggestions on my research and personal growth.

To my friends at UT Austin, thank you for the many joyful memories with you. The support I received from my family can never be overstated. I owe my deepest gratitude to my husband, Siyuan Lu, for being on my side. His incredible understanding, support, comfort, encouragement, and love helped me through the most stressful times.

# **Models and Methods for Operational Planning in Freight Railroads**

Xiaoyan Si, Ph.D.

The University of Texas at Austin, 2015

Supervisors: Jonathan F. Bard, Anantaram Balakrishnan

Railroads are facing increasing demand for freight transportation. Effective planning and scheduling are crucial to improve the utilization of expensive resources (such as crew and track), reduce operational costs, and provide on-time service. This dissertation focuses on problem modeling and solution method development for real planning problems faced by railroads. It consists of three chapters that study two important planning problems in the daily operations of U.S. freight railroads: crew assignment and train movement planning. Chapter 2 proposes an optimization model to decide crew-to-train assignments and deadheads for double-ended crew districts. We develop an effective solution approach, combining optimization and a standalone heuristic, that generates optimal solutions in minutes. The excellent performance of this solution approach makes it well-suited for implementation within a real-time decision support tool for crew dispatchers. Chapter 3 discusses crew repositioning given the uncertainty in trains' arrival and departure times. We propose models that minimize the expected crew holding, train delay, and deadheading cost, and develop both exact and heuristic solution methods to provide insights for crew planning under train schedule uncertainty. The last chapter studies the movement planning problem for trains traveling in a territory with multiple through tracks (mainlines) and various junctions.

We explore a number of heuristic algorithms to obtain good solutions within a reasonable amount of time. The contributions of this dissertation include modeling enhancements, algorithmic development, implementation and computational testing, and validation using real data.

## Table of Contents

List of Tables .....	xi
List of Figures .....	xii
Chapter 1: Introduction .....	1
Chapter 2: Real-Time Crew Assignment in Double-Ended Districts .....	5
2.1 Introduction.....	5
2.2 Literature Review.....	8
2.3 Crew Assignment Problem in PSQ districts .....	12
2.3.1 Problem Ingredients .....	12
2.3.2 Crew Dispatching Rules in PSQ Districts .....	14
2.4 Model Formulation .....	18
2.4.1 Mixed-Integer Program.....	19
2.4.2 Constraints for PSQ Crew Dispatching Rules .....	23
2.4.2.1 Rotation Key constraints.....	25
2.4.2.2 Push-Pull constraints.....	27
2.5 Model Enhancements.....	31
2.5.1 Deadhead Capacity Cuts .....	31
2.5.2 Stronger Rotation Key Constraints .....	32
2.5.3 Problem Reduction: Latest Connection .....	34
2.6 Crew Assignment Heuristic .....	38
2.7 Computational Results .....	41
2.7.1 Performance Analysis .....	42
2.7.2 Computational Results by Crew District .....	48
2.7.3 Impact of Problem Parameters.....	50
2.7.3.1 Capacity of secondary queue .....	50
2.7.3.2 Deadheading cost .....	52
2.8 Conclusions.....	54



Chapter 3: Crew Planning with Uncertainty in Train Arrival and Departure Times	56
3.1 Introduction.....	56
3.2 Literature Review.....	58
3.3 Problem Description .....	62
3.4 Problem Modeling .....	65
3.5 Model 1: Inbound Deadheading.....	68
3.5.1 Recursive method for inbound deadheading.....	69
3.5.2 Newsvendor-based heuristic for inbound deadheading .....	70
3.5.2.1 Optimal solution for a continuous multi-period inventory problem .....	71
3.5.2.2 Base heuristic for problem with discrete supply and demand .....	77
3.5.2.3 Modified heuristic for problem with discrete supply and demand.....	79
3.6 Model 2: Bi-directional Deadheading.....	81
3.6.1 Recursive method for bi-directional deadheading .....	82
3.6.2 Newsvendor-based heuristic for bi-directional deadheading.....	83
3.7 Computational Experiments.....	87
3.7.1 Distribution of base train schedule .....	88
3.7.2 Train schedule deviation .....	89
3.7.3 Distribution of cumulative supplies and demands .....	91
3.7.4 Testing instances .....	93
3.7.5 Compare solution methods of model [IBCP].....	94
3.7.5.1 Performance comparison with default parameters.....	95
3.7.5.2 Sensitivity to cost parameters .....	101
3.7.6 Compare solution methods for model [IOCP] .....	108
3.7.6.1 Performance comparison with default parameters.....	108
3.7.6.2 Sensitivity to cost parameters and schedule variability	111
3.8 Conclusions.....	115
Chapter 4: Train Movement Planning.....	119
4.1 Introduction.....	119

4.2 Literature Review.....	121
4.3 Problem Description .....	126
4.4 Model Formulation .....	129
4.4.1 Movement Planning Formulation .....	131
4.4.2 Train-based forcing constraints.....	133
4.4.3 Headway-based forcing constraints .....	135
4.5 Optimization-Based Heuristics .....	136
4.5.1 LP-based time window method .....	137
4.5.2 Decomposition by train groups .....	143
4.5.3 Rounding heuristic .....	145
4.5.4 Directional movement planning.....	147
4.5.5 Expanding planning horizon .....	149
4.6 Sequential Conflict Resolution Heuristic.....	152
4.6.1 Construct an initial solution .....	152
4.6.2 Train re-routing.....	160
4.7 Computational Results .....	161
4.7.1 Base Model .....	163
4.7.2 LP-based heuristics .....	164
4.7.3 Directional movement planning.....	174
4.7.4 Expanding planning horizon .....	175
4.7.5 Sequential conflict resolution heuristic.....	177
4.8 Conclusion .....	180
Appendix A: Forcing Constraints of Indicators.....	182
References.....	183

## List of Tables

Table 2.1 Variations of Rotation Key constraints based on constraints (2.10a)....	34
Table 2.2 Problem size of selected instances and model size of the Base model..	44
Table 2.3 Model size reduction and LP bound improvement: Models $S^3$ and $S+R^4$	44
Table 2.4 Computational performance comparison: Base, Model S+R, and Final	46
Table 2.5 Average problem size and computational results for 14 crew districts.	49
Table 2.6 Example: impact of secondary queue capacity .....	51
Table 2.7 Impact of secondary queue capacity on CPU time and heuristic gap....	52
Table 2.8 Example: impact of deadheading cost .....	53
Table 2.9 Impact of changing deadheading costs by 30% .....	54
Table 3.1 Testing groups and base schedule distributions.....	94
Table 3.2 Three heuristics for [IBCP]: percentage gap of total cost .....	96
Table 3.3 Heuristics for [IOCP]: percentage gap of total cost.....	109
Table 4.1 Base model problem size and model size .....	163
Table 4.2 Base model computation result summary .....	165
Table 4.3 LP-based time window heuristic: time window and integral path.....	167
Table 4.4 LP-based time window heuristic: model size and computation results	168
Table 4.5 Characteristics and performance of decomposition by train groups....	170
Table 4.6 LP solution with and without headway-based forcing constraints .....	173
Table 4.7 Performance of rounding heuristic .....	174
Table 4.8 Result of directional movement planning .....	176
Table 4.9 Result summary for expanding horizon heuristic .....	178
Table 4.10 Solution of different conflict resolution strategies .....	179
Table 4.11 Solution summary for all methods.....	180

## List of Figures

Figure 2.1 Illustration of PSQ dispatching rule .....	16
Figure 2.2 Example of crew queue dynamics and assignments for PSQ districts .	18
Figure 2.3 Stronger Rotation Key constraints with unit capacity trip .....	33
Figure 2.4 The latest-connection identification procedure .....	37
Figure 2.5 Flow chart of the crew assignment procedure .....	38
Figure 3.1 Distribution of base train schedule .....	89
Figure 3.2 Two Gamma distributions of train schedule deviation.....	90
Figure 3.3 Gap of expected crew holding cost .....	99
Figure 3.4 Gap of expected train delay cost .....	99
Figure 3.5 Gap of average supply level .....	100
Figure 3.6 Gaps of deadhead count and average deadheading time .....	100
Figure 3.7 Base_NV: percentage gap of total cost vs non-monotonicity of $X^*$ ...	101
Figure 3.8 Percentage gap of total cost .....	101
Figure 3.9 Performance sensitivity to crew holding cost (IBCP) .....	104
Figure 3.10 Performance sensitivity to train delay cost (IBCP) .....	105
Figure 3.11 Performance sensitivity to deadheading cost (IBCP) .....	107
Figure 3.12 Deterministic_schedule: gaps of deadheading counts and time .....	110
Figure 3.13 Deterministic_schedule underestimates crew holding and train delay	111
Figure 3.14 Performance sensitivity to crew holding cost (IOCP) .....	112
Figure 3.15 Performance sensitivity to train delay cost (IOCP) .....	115
Figure 3.16 Performance sensitivity to schedule variance (IOCP) .....	117
Figure 4.1 Track layout.....	128
Figure 4.2 Illustration of train-based forcing constraints.....	134

Figure 4.3 Illustration of headway-based forcing constraints.....	136
Figure 4.4 Illustration of LP-based delay estimation.....	142
Figure 4.5 Construct initial solution .....	154

## **Chapter 1: Introduction**

Railroads are vital to the economic well-being of a country. In the U.S., railways transport about 40% of all freight (measured in ton-miles) every year (FRA 2010). The total shipments by rail are expected to reach 28.5 billion tons in 2040, representing a 45% increase compared to 2012 (AAR, 2015(a)). The railroad industry is capital intensive. Since 2012, freight railroads spent over \$25 billion per year to maintain and upgrade their infrastructure (such as tracks, yards, and traffic signals) and equipment (such as locomotives and freight cars) (AAR, 2015(a)). Operating a railroad is also very expensive. Besides fuel cost, crew wages and benefits account for a large proportion of total operating expenses. In 2013, the three largest U.S. freight railway companies each spent an average of \$547 million on train crew wages, accounting for 11% of their train operating costs (STB, 2013). Given the rapidly growing demand for rail freight transport services and high operating expenses, effective utilization and deployment of resources such as crews and tracks is very important for railroads to operate profitably and provide on-time service. In this dissertation, we develop models and methods to assist effective planning of crews and track usage in freight railroads.

Railways are highly complex systems that involve a rich set of planning and scheduling problems. These problems naturally fall into the three hierarchical planning levels of strategic, tactical, and operational planning. Strategic planning entails long-term decisions such as the design and expansion of the rail network, yard location and capacity planning, locomotive acquisition, and fleet sizing. The tactical stage involves

various medium term planning problems, such as human resource (e.g., crews) management, deciding train routes and frequencies, developing blocking policies, and scheduling preventive maintenance activities for tracks and equipment. Finally, the operational stage deals with planning daily and weekly operations, including planning the detailed movements of trains in each territory, assignment of crew members to trains, assignment of locomotives to power the trains, and railcar loading and unloading operations in yards. Nemani and Ahuja (2010) provide an overview of planning problems in freight railroads and optimization models for several of these problems.

Unlike passenger railways, the train schedules for freight trains are not periodic, but can vary from week to week. Consequently, operational planning problems for freight railroads are more challenging and require frequent and quick decision-making. In this dissertation, we study two important operational planning problems – crew assignment and train movement planning – for a given set of (desired) train schedules in the near term (few hours to few days). The goal of this dissertation is to develop optimization tools that can assist crew planners and train dispatchers in developing crew assignment and deadheading plans and deciding train movements.

This dissertation consists of three chapters. Chapter 2 presents our work on real-time crew assignment in double-ended crew districts with complex crew dispatching rules. Given the schedule of trains passing through a crew district, we develop an optimization model to assign crew members to trains and deadhead crews, as needed, so as to avoid crew shortages and train delays while minimizing total crew costs, including deadheading and crew holding costs. Computational testing and validation using 140

real-life problem instances demonstrates that our solution approach, combining an exact solution procedure with a heuristic algorithm, is very effective and solves most of these instances within one minute.

Chapter 3 considers crew repositioning (deadheading) in a longer term (a few days) and takes into account the uncertainty in train arrival and departure times. Past research on train crew scheduling (including our study in Chapter 2) assumes that train schedules are deterministic, i.e., known with certainty, and focuses on decision support. The models and analysis discussed in this chapter are intended to serve as the first steps for incorporating uncertainty in train arrival/departure time for railway crew planning, and to identify effective crew availability and deadheading policies. We propose models that balance the costs for crew holding, train delay (due to crew unavailability), and deadheading, given the probability distribution for train arrival and departure times. We develop a dynamic programming algorithm to solve the problem of deadhead planning, taking into account linear crew holding and train delay costs, and deadheading cost per crew member. Further, we propose several heuristics that are based on principles underlying the newsvendor policy for inventory management (i.e., ordering up to a critical fractile that depends the relative magnitudes of overage and underage costs), but extended to a multi-period setting. Our analysis and computational results provide insights of deadhead planning policy with uncertainty in train arrival and departure times.

Chapter 4 considers the problem of planning train movements in a dispatching territory with multiple mainlines and various junctions, i.e., sidings, and crossovers. Given a set of trains entering a territory in a given planning horizon and the layout of



tracks in the territory, the goal is to determine the timing and sequence of each train's movements through the segments of the territory so as to minimize the total (weighted) delay of trains and avoid conflicts between train movements. We propose a discrete-time integer programming model that optimizes the segment-by-segment movements of trains and incorporates all safety requirements for a conflict-free solution. To obtain good solutions in a reasonable time, we explore multiple methods that either heuristically reduce the size of the optimization model or impose integrality constraints iteratively. In addition, we develop a standalone heuristic that can provide a feasible solution within seconds.

As the preceding discussions suggest, the research topics in this dissertation are motivated by real planning and scheduling problems faced by railroad companies. The contributions of this dissertation include: (1) problem modeling that incorporates various regulations and operational rules; (2) strengthening the model to accelerate solution procedures; (3) methodological developments that focuses on generating good solutions quickly, permitting their practical use for real-world decision support; (4) implementation and computational testing to assess the performance of solution approaches and provide insights; and, (5) solution validation using real-life/real-size data.

## **Chapter 2: Real-Time Crew Assignment in Double-Ended Districts**

### **2.1 INTRODUCTION**

For freight railroad companies in U.S., crew costs account for the second largest portion of train operating expenses. Therefore, effective deployment and utilization of train crews is a very important priority for railroads. Unlike the airline industry, which has widely adopted and implemented optimization-based decision support systems for crew assignment and scheduling, the freight railroad industry largely relies on manual decision-making for crew assignment. This situation stems in part from the fact that the models and methods for airline crew scheduling do not apply to freight railroads because of the many differences in the structure and operational rules for crew deployment in these two settings. Prior research on optimization models for train crew scheduling considers only simple crew dispatching rules (such as the First-In-First-Out rule), whereas double-ended districts in which crews are based at more than one station require more sophisticated rules that the literature does not address. This study formulates a large-scale mixed-integer programming model and develops effective solution procedures to support short-term crew assignment decisions for double-ended districts with dispatching rules based on primary-secondary queues.

Unlike passenger transportation services (such as passenger rail, airlines, and buses) that follow periodic schedules, freight trains have ad hoc schedules that vary daily based on the current freight transportation demand between various origin-destination pairs. Consequently, crew assignment for freight railroads is an ongoing short-term

planning requirement to decide which crew members to assign to every train that is scheduled to run each day. Trains require crew members from different “occupations”, e.g., engineer and conductor. For crew planning purposes, the rail network is partitioned into crew districts, each demarcated by two stations at which train crews change for all trains traversing the district in either direction between these stations. We address the crew assignment problem for double-ended districts in which both stations serve as home bases for crews. Each station also has extra crew members who can be assigned to a trip when no regular crew member is available. Thus, for each occupation, there are several categories or *pools* of crews (distinguished by their home base and whether they are regular or extra crews) from which to select crew members for every trip. Crew planners must make two main sets of decisions: which crew members (i.e., from which pool for each occupation) to assign to each scheduled train passing through the district, and whether and when to deadhead crew members from one station to the other using what mode of transport. Crew deadheading may be needed to relieve crew shortages or reduce crew layover times and costs; deadheading modes include using spare seats on regular train trips, public transportation, and taxis. To ensure equitable workload distribution within and across crew bases, the crew assignment decisions in each district must follow certain dispatching rules that are specified by labor union agreements. In this study, we focus on crew deployment in districts that use the so-called *primary-secondary-queue* (abbreviated as *PSQ*) discipline to regulate the assignment of crews from the two home bases. These rules, applicable to several major U.S. railroads and mandated by long-standing agreements with labor unions, require

maintaining two lists or queues of crew members at each station, and govern the composition, ordering, and updating of these lists as crews arrive or depart (Section 2.3.2 provides a detailed description of this discipline). In this setting, given the schedule of all trains that will traverse the district during the short-term planning horizon, the crew assignment problem entails deciding deadheading plans and crew-to-trip assignments, subject to rest requirements and the dispatching rules applicable to the district, so as to minimize total costs, consisting of crew assignment, deadheading, and layover costs. We formulate an appropriate optimization model for this problem, develop an effective solution procedure to solve it quickly and optimally (or near-optimally, with performance guarantees) so as to support real-time crew planning, and apply this model and method to real problem instances.

This research provides three broad contributions to solve practical crew scheduling problems in U.S. freight railroads. First, we provide a new model formulation for crew assignment in PSQ districts; the complicated dispatching rules for such districts have not been previously modeled or solved in an optimization framework. The PSQ rules, fixed by agreements with labor unions, apply to crew districts in several major U.S. railroads, and so our model and method have broad appeal. Second, we propose several model enhancements and solution techniques that exploit the problem structure, including model strengthening, problem reduction, and a heuristic procedure, to solve the problem effectively. Our modeling improvements and solution approach also extend (possibly with minor modifications) to other crew districts that use alternate dispatching rules. Third, we discuss the results from testing the model and solution

approach using actual data. We validated our model and solution approach using 140 real-life problem instances provided by a leading U.S. freight railroad. The computational tests confirm that our modeling enhancements and heuristic algorithm significantly improve solution performance. Specifically, the approach generates high-quality heuristic solutions in a few seconds, and optimally solves most problem instances within one minute. In contrast, for a sample of seven problem instances, a commercial solver (CPLEX) applied to a base model without enhancements for up to one hour of computational time did not even yield any feasible solution for four instances and had gaps of 2.91% and 7.35% at termination for two other instances. The excellent computational performance of our approach makes it practical for real-time use to support crew planning decisions and conduct timely what-if analyses. The company plans to implement this model and methodology within its suite of tools for operations planning.

The rest of this chapter is organized as follows. Section 2.2 briefly reviews the literature on related crew planning problems. Section 2.3 describes the ingredients of the crew assignment problem in double-ended districts, and Section 2.4 presents our mixed-integer programming model formulation incorporating the PSQ dispatching rules. Sections 2.5 and 2.6 discuss model enhancements and the heuristic algorithm, respectively. We present extensive computational results in Section 2.7, and offer concluding remarks in Section 2.8.

## **2.2 LITERATURE REVIEW**

A significant portion of the research literature on transportation crew planning

focuses on the airline industry (e.g., see Barnhart et al. (2003) and Gopalakrishnan and Johnson (2005) for reviews of airline crew scheduling models and methods). In the railway context, most prior work deals with crew scheduling for passenger railways. As with airlines, passenger trains follow periodic schedules that do not change frequently. A common approach for both airlines and passenger railroads is to decompose crew scheduling into two phases: crew pairing and crew rostering. The crew pairing problem generates and selects feasible pairings, i.e., sequences of duties (work during a day) that start and end at the same crew base, so that all scheduled trips are covered. The crew rostering problem then combines the chosen pairings into rosters and assigns them to individual crew members. This stream of research models the crew pairing problem as a variant of the set partitioning problem where each column corresponds to a pairing. Due to the exponential number of feasible pairings, several researchers have applied column generation methods to solve these problems (e.g., Freling et al. 2004, Abbink et al. 2005). Caprara et al. (1997, 1998, 2001, 2007) discuss different modeling approaches and solution methods for crew pairing and rostering in European passenger railways. For some freight railroads outside the U.S. (e.g., Australia and Germany) whose operational rules are similar to those for passenger trains, some researchers (e.g., Ernst et al., 2001, Jütte et al., 2011) have adapted modeling and solution methodologies from passenger railway crew scheduling.

The traditional modeling and solution approaches for crew pairing and rostering do not apply to the problem we study because of many differences in the structure and operation of these settings. First, the schedules for freight trains in the U.S. are ad hoc

rather than periodic, necessitating short-term crew assignment decisions, whereas airlines and passenger railways operate on stable and cyclic schedules. Second, the available crew members belong to different crew pools, and planners must decide which crew pool to use for each trip. Further, since crew pools differ in their costs and/or deployment rules, we cannot separate the crew pairing and rostering decisions into sequential stages. Third, the operational rules in U.S. freight railroads are different from those in other transportation contexts. Crew assignment for freight trains must follow complicated crew dispatching rules that are not easy to model within the framework of conventional set partitioning formulations that define columns based on pairings. Finally, since freight crew assignment decisions decompose by district and a crew member's duty is just one trip from one station to the other (for double-ended districts), each pairing consists of two consecutive trips going back and forth between the two stations; due to this simpler structure, the number of possible pairings is polynomial in the number of trips in that district during the planning horizon.

We next review the few optimization-based papers that are related to the freight train crew scheduling setting that we study. Şahin and Yüceoğlu (2011) discuss crew assignment in Turkish State Railways for regions with one home station and multiple away stations; trains only run between the home station and each away station. To incorporate the requirement that each crew member has to take one calendar day off every week, they formulate and solve an optimization model based on a layered space-time network representation and also propose a two-stage heuristic that first solves a minimum flow problem to find the minimum number of crew members needed without

the day-off requirement, and then incorporates this requirement heuristically. For U.S. freight railroads, Gorman and Sarrafzadeh (2000) propose a heuristic algorithm to schedule crews in single-ended districts (i.e., districts in which all crew members are based at only one of the two stations in the district), assuming a particular crew dispatching rule at each station and considering only regular crews (no extra crews). They first apply a dynamic programming algorithm to solve a restricted problem that limits crew deployment options and rest requirements, and then heuristically improve the solution by relaxing these restrictions. Vaidyanathan et al. (2007) discuss a multi-commodity network flow model approach for crew scheduling in North American railroads, but consider only the First-In-First-Out (FIFO) rule for crew assignments. They propose two algorithms, successive constraint generation (SCG) and quadratic cost perturbation (QCP), to solve the problem. The SCG method starts by solving the relaxed model without any FIFO constraints, and iteratively adds these constraints to remove FIFO infeasibilities in the intermediate solutions while the QCP method uses an arc cost perturbation scheme (without explicit FIFO constraints) to ensure feasibility.

For double-ended districts, modeling the additional or alternative crew dispatching rules (beyond FIFO) is important both to accurately capture the cost of crew assignment (e.g., crew layover cost) and yield implementable solutions in practice. Crew dispatching rules, especially the PSQ rules we study, are not easy to formulate and cannot be handled with just cost perturbations. We build upon the network flow modeling approach over a time-space network by adding constraints to enforce the crew dispatching rules. We can also interpret our model as a formulation with all feasible



pairings since each pairing in the railway crew scheduling context consists of two trips, one in each direction. The additional crew dispatching rules complicate the crew assignment model and make it difficult to solve; we, therefore, propose a tailored solution approach. Next, we formally describe the crew assignment problem for double-ended districts, and elaborate on the PSQ dispatching rule.

### **2.3 CREW ASSIGNMENT PROBLEM IN PSQ DISTRICTS**

Given the schedule of trains that will traverse a crew district in the next few days, the crew dispatcher must select crew members of required occupation types (engineer, conductor, and brakeman) from the available crew pools to operate each train and make deadheading decisions so as to minimize the total cost for crew-to-trip assignments, deadheading, and layovers while meeting all crew assignment restrictions and policies. We refer to each freight train that is scheduled to pass through the district as a “regular” trip (distinguished from candidate “deadhead” trips), and assume that these trips cannot be delayed or canceled. We next discuss the features and relevant costs to consider when making crew assignment decisions, and later (in Section 2.3.2) elaborate on the crew dispatching rules for PSQ districts.

#### **2.3.1 Problem Ingredients**

The important ingredients of the crew assignment problem in double-ended districts are the choices of available crews, rest requirements and layover costs, and deadheading options.

**Crew pools:** A double-ended district has two crew changing stations, one at either end, with each station serving as the home base for a subset of crew members assigned to

that district. Further, every home base has two types of crews for each occupation type: regular crews and extra crews. We refer to the different categories of crews, distinguished by their home station and whether they are regular or extra crews, as different crew pools. The costs and dispatching rules can vary by pool. Extra crews should be used sparingly and only when no regular or away crew is rested and available to operate a trip, and their work ends as soon as they return home after completing a round trip.

**Rest requirements and layovers:** Since double-ended districts are fairly long, traversing the district can take more than half a day. Therefore, every crew member can operate or take at most one trip in a day, i.e., a duty consists of a single trip, and must rest for at least a minimum specified time (e.g., 10 hours) before the next trip. For crews resting at their home station, the company does not incur any cost. But, at the away station, the company must pay a daily lodging cost for each crew member. Further, if the layover time at the away station exceeds a specified *heldaway* threshold (e.g., 16 hours), the company must pay a per-hour heldaway fee to that crew member for the amount of time exceeding this threshold. We refer to the lodging cost plus any heldaway fee at the away station as the *layover* cost.

**Deadheading options and costs:** Deadheading refers to transporting a crew member from one station to another without assigning him/her the operational responsibility for a trip. Crews may be deadheaded either to relieve shortage of crew members at a station due to traffic imbalance in the two directions or to avoid excessive layover time (and consequent layover cost) at the away station. Thus, deadheading

decisions depend on crew availability and layover times at both stations, as well as the costs for deadheading versus using extra crews. Such decisions are further complicated by the many available options for deadheading in terms of transportation modes and timing. The three most common modes for deadheading are: regular trains that have extra seats (in the locomotive), public transportation (e.g., buses, passenger trains), and taxis. These transportation options vary in their costs, speed, capacity, and timing. All crew members who are deadheaded must be paid a *trip rate* (typically the same amount paid to crews who operate a train). In addition, the company incurs a transportation fare for crews deadheaded on public transportation, and a fixed cost for each taxi. Regular trains have limited number of additional seats (that can vary by train) for deadheading crew members, and each taxi can carry no more than a specific number of passengers. So, different deadheading modes have different costs and capacities. Moreover, trains and public transportation are available only at fixed times, whereas taxis can be scheduled as needed.

### **2.3.2 Crew Dispatching Rules in PSQ Districts**

When assigning crews to trips (regular or deadhead), the choice of crew pool and specific crew member from a pool must follow the dispatching rules that apply to the crew district. The dispatching rules in PSQ districts are based on the state of two queues, called the **primary queue** and **secondary queue**, at each station for every occupation. A crew member arriving at his/her away station joins the primary queue at that station, whereas a crew member returning to his/her home station goes to the secondary queue. The primary queue can hold an unlimited number of crew members,

but the secondary queue has a fixed capacity. Whenever the number of home crew members in the secondary queue exceeds its capacity, the first crew member in this queue will be “*pushed*” to the primary queue, thus blending the home and away crews in the primary queue.

The order in which crew members join each queue, and hence their dispatching order, depends on the *rotation key* specified for the district and occupation. The rotation key is either First-In-First-Out (FIFO) or First-Out-First-Out (FOFO). For the FIFO rule, the relative dispatching order among crews from each home base is the same as the order of arrivals, and so crews go to the end of the appropriate (primary or secondary) queue upon their arrival. On the other hand, with the FOFO rule, crew members who departed earlier on their preceding (inbound) trips receive higher dispatching priority. So, even if a trip  $i$  overtakes or crosses another trip  $i'$  in the same direction (i.e., trip  $i'$  starts earlier but ends later than trip  $i$ ), the crew on trip  $i'$ , when rested, must be dispatched earlier than any crew member from the same home base who travels on trip  $i$  even though trip  $i'$  arrives later.

Figure 2.1 depicts how arriving crew members join the primary or secondary queue at a station, and how crew members in the secondary queue get pushed to the primary queue. The away crew members A1, A2, and A3 join the primary queue, while the home crews B1 to B4 go to the secondary queue upon arrival. When B4 joins the secondary queue, the capacity (equal to three, in this example) of this queue is exceeded, and so B1 is “*pushed*” from the secondary queue to the primary queue. Thus, the pre-specified capacity parameter for the secondary queue, which can vary by station and

occupation type, effectively serves to interleave and balance the assignment of workload among crews based at the two stations.

For every outbound trip (regular or deadhead) departing from a station and for each occupation, the dispatcher must assign the first rested crew member from the primary queue for that occupation. If none of the crews in the primary queue is rested (i.e., their current layover time is less than the minimum required rest time), then the first rested crew member from the secondary queue is “*pulled*” to the primary queue and assigned to that trip. Further, if no crew member in the secondary queue is rested, the dispatcher must assign an extra crew member homed at that station.

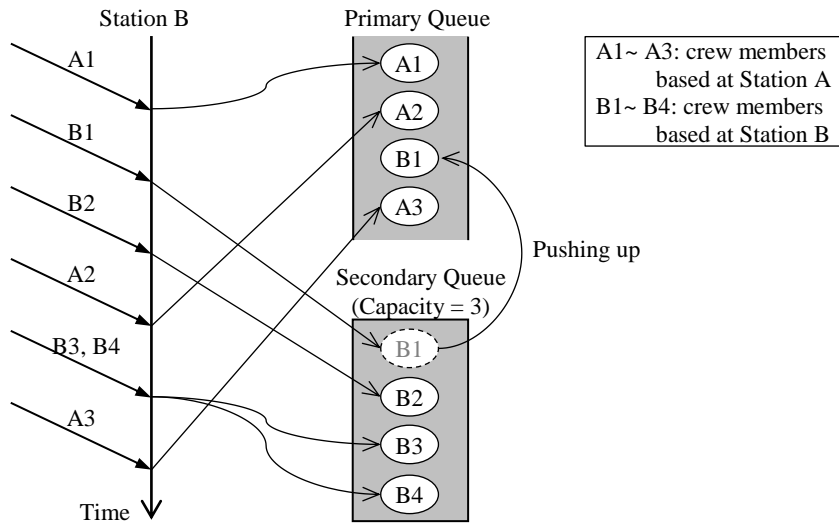
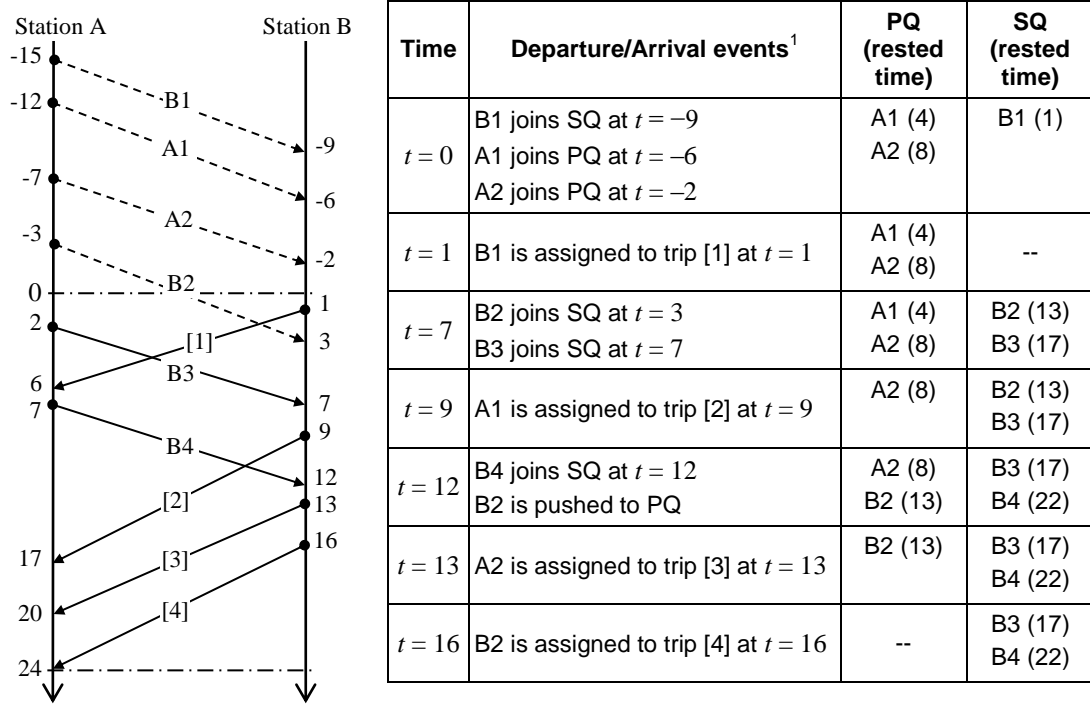


Figure 2.1 Illustration of PSQ dispatching rule

Next, we illustrate, using a simple example, the assignment decisions and queue dynamics for the PSQ scheme. Figure 2.2(a) pictorially depicts the schedule of trips and crew assignments, and Figure 2.2(b) lists the corresponding queue update steps. In

Figure 2.2(a), the two vertical lines represent the two stations A and B, and the y-axis depicts the time of various arrival and departure events at each station. The two horizontal dashed lines represent the start ( $t = 0$ ) and end ( $t = 24$ ) time of the planning horizon, respectively. The dashed arcs between the two stations are initial trips that were assigned before time zero, whereas the solid arcs are future trips that require crews. We focus on the primary and secondary queue dynamics for a particular occupation (e.g., engineers) at Station B and the assignment of crews to the B-to-A trips (numbered as [1], ... [4]), assuming for convenience that crews from appropriate pools have already been chosen for the A-to-B trips (the identity of the crew member assigned to each A-to-B trip is shown on the arc corresponding to the trip). Suppose the secondary queue at Station B has a capacity of two, the required minimum rest time between trips is 10 hours, and the rotation key at Station B is FIFO. Figure 2.2(b) shows the states and transitions of the two crew queues at station B at time instances when crews arrive or depart from station B. At time  $t = 1$ , home crew B1 is pulled up and assigned to trip [1] since A1 and A2 who are currently in the primary queue will not be fully rested until  $t = 4$  and  $t = 8$ , respectively. Next, A1 is assigned to trip [2] at  $t = 9$ . B2 is pushed to the primary queue at  $t = 12$  when B4 joins the secondary queue since the secondary queue at station B can only hold up to two crew members. Finally, A2 is assigned to trip [3] and B2 is assigned to trip [4]. This example illustrates how the pushing and pulling processes of the PSQ mechanism interleave the assignments to home and away crews, thus distributing the workload equitably between the two home bases.



<sup>1</sup>PQ = Primary Queue, SQ = Secondary Queue

Figure 2.2 Example of crew queue dynamics and assignments for PSQ districts

## 2.4 MODEL FORMULATION

The main decisions for the crew assignment problem are which available crew member to assign to each trip, and when and how many crew members of each occupation to deadhead using which deadhead mode. Since the layover costs depend on the time that a crew member spends at his/her away station between trips, we require variables to capture the “connection,” i.e., the consecutive inbound and outbound trips, for each crew member. A connection from an inbound trip  $i$  to an outbound trip  $j$  is feasible if the time from the end of trip  $i$  to the start of trip  $j$  equals or exceeds the minimum rest time (unless  $j$  is the dummy sink trip representing the end of the planning

horizon). Each crew member must operate at least one regular trip during their round-trip tour or pairing, and so we do not permit deadhead-to-deadhead connections. Since assignment, layover, and deadheading costs depend only on occupation and crew pool rather than the identity of the specific crew member, our formulation uses connection variables for each occupation and pool instead of each individual crew member. In practice, each regular trip requires one crew member of each occupation; further, we assume without loss of generality that each initial trip (assigned before time zero) supplies at most one crew member of each occupation (we can split any initial trip, e.g., taxi, that carries multiple crew members into multiple initial trips). Therefore, we can treat the connection variables as binary variables. Finally, we assume that an unlimited number of extra crew members are available, but the company incurs a high per-person cost for utilizing an extra crew member.

### 2.4.1 Mixed-Integer Program

In this section, we develop a mixed-integer program for our crew assignment problem (CAP). For this purpose, we define the following notation:

#### Sets and indices

$S$  : set of **stations**,  $S = \{A, B\}$ , indexed by  $s$

$K$  : set of **occupation** types, indexed by  $k$

$P$  : set of **crew pools**, indexed by  $p$

$P_s$  : set of **crew pools based at station  $s$** ;  $PX_s$  is the index of the extra pool at station  $s$

$I, R$  : set of **initial** and **regular trips**

$D$  : set of candidate **deadhead** trips, consisting of *regular train deadhead* trips ( $DR$ ), *public transit* trips ( $DP$ ), and candidate *taxis* ( $DT$ ),  $D = DR \cup DP \cup DT$

$Q$  : set of **all trips** in the planning horizon,  $Q = I \cup R \cup D$

$IJ$  : set of feasible **inbound-to-outbound trip connections**



$sd_i, sa_i$  : **departure** and **arrival stations** for each trip  $i \in Q$

$td_i, ta_i$  : **departure** and **arrival times** for each trip  $i \in Q$

### Parameters

$W_i^{kp}$  : number of crew members of occupation  $k$  from pool  $p$  arriving on initial trip  $i$ ,

$$W_i^{kp} \in \{0,1\}$$

$U_i$  : capacity of deadhead trip  $i \in D$

$\beta_s^k$  : capacity of the secondary queue for occupation  $k$  at station  $s$ ,  $\beta = \{\beta_s^k : k \in K, s \in S\}$

$C_{ij}^{kp}$  : layover costs, including lodging and possible heldaway cost, for a crew member of occupation  $k$  from pool  $p$  connecting from trip  $i$  to trip  $j$ , for all  $\langle i, j \rangle \in II$ ,  $k \in K$  and  $p \in P$

$E_i^{kp}$  : assignment cost, including trip rate, public transit fare (if  $i \in DP$ ) and extra crew cost (if  $p$  is an extra pool at the departure station of trip  $i$ ), for assigning a crew member of occupation  $k$  from pool  $p$  to trip  $i$ , for all  $i \in Q \setminus I$ ,  $k \in K$  and  $p \in P$

$F_i$  : fixed cost of taxi trip  $i \in DT$

$M$ : a sufficiently large positive number

The formulation uses three types of variables: *connection variables*  $x$ , *taxi selection variables*  $y$ , and *assignment variables*  $v$ . Specifically, let:

$x_{ij}^{kp} = 1$  if a crew member of occupation  $k$  from pool  $p$  is transferred from trip  $i$  to trip  $j$ ,

and 0 otherwise, for all  $\langle i, j \rangle \in II$ ,  $k \in K$  and  $p \in P$ ;

$y_i = 1$  if taxi trip  $i$  is selected in the solution, and 0 otherwise, for all  $i \in DT$ ; and,

$v_i^{kp}$  = number of crew members of occupation  $k$  from pool  $p$  assigned to trip  $i$ , for all  $i \in Q$ ,

$k \in K$  and  $p \in P$ .

The assignment variables  $v_i^{kp}$  are not essential for our model since they are implied by the connection variables  $x_{ij}^{kp}$ ; however, including these variables helps to

reduce the density of the constraint matrix and makes the formulation easier to follow. Due to the complexity of the crew dispatching rules in PSQ districts, we first represent these rules as a general discrete set of feasible assignments  $\mathbf{X}(RK, \beta)$ , defined as the set of crew assignments that satisfy the PSQ crew dispatching rules, based on applicable rotation keys (RK) and secondary queue capacities ( $\beta$ ). Later (in Section 2.4.2), we discuss how to represent these rules as linear constraints.

Using the above decision variables and parameters, we can formulate the crew assignment problem as the following mixed-integer program, denoted as [CAP]:

$$[\text{CAP}] \text{ Minimize } \sum_{\langle i,j \rangle \in IJ} \sum_{k \in K} \sum_{p \in P} C_{ij}^{kp} x_{ij}^{kp} + \sum_{i \in Q \setminus I} \sum_{k \in K} \sum_{p \in P} E_i^{kp} v_i^{kp} + \sum_{i \in DT} F_i y_i \quad (2.1)$$

subject to:

Crew requirement:

$$\sum_{p \in P} v_i^{kp} = 1 \quad \forall i \in R, k \in K, \quad (2.2)$$

Flow conservation:

$$v_i^{kp} = W_i^{kp} \quad \forall i \in I, k \in K, p \in P, \quad (2.3a)$$

$$v_j^{kp} = \sum_{i: \langle i,j \rangle \in IJ} x_{ij}^{kp} \quad \forall j \in Q \setminus I, k \in K, p \in P, \quad (2.3b)$$

$$v_i^{kp} = \sum_{j: \langle i,j \rangle \in IJ} x_{ij}^{kp} \quad \forall i \in Q, k \in K, p \in P, \quad (2.3c)$$

Deadhead capacity:

$$\sum_{k \in K} \sum_{p \in P} v_i^{kp} \leq U_i y_i \quad \forall i \in DT, \quad (2.4a)$$

$$\sum_{k \in K} \sum_{p \in P} v_i^{kp} \leq U_i \quad \forall i \in DR \cup DP, \quad (2.4b)$$

PSQ dispatching rules:

$$x \in \mathbf{X}(RK, \beta), \quad (2.5)$$

Extra pool rule:

$$\sum_{\substack{i: \langle i, j \rangle \in IJ \\ j': \langle i, j' \rangle \in IJ, td_{j'} > td_j}} \sum_{p' \in P \setminus \{p\}} x_{ij'}^{kp'} \leq M(1 - v_j^{kp}) \quad \forall j \in R, k \in K, p = PX_{sd_j}, \quad (2.6)$$

Nonnegativity and integrality:

$$x_{ij}^{kp} \in \{0, 1\} \quad \forall \langle i, j \rangle \in IJ, k \in K, p \in P, \quad (2.7)$$

$$v_i^{kp} \geq 0 \quad \forall i \in Q, k \in K, p \in P, \quad (2.8)$$

$$y_i \in \{0, 1\} \quad \forall i \in DT. \quad (2.9)$$

The objective function (2.1) minimizes the total crew costs, including layover, assignment, and taxi fixed costs. Constraints (2.2) ensure that every scheduled (regular) train is assigned one crew member of each occupation type. Constraints (2.3a) to (2.3c) express the assignment variables in terms of the initial trip assignments and connection variables, and ensure that the number of crew members of occupation  $k$  from pool  $p$  transferred to each trip  $i$  at the starting station equals the number of crew members transferred from this trip at the ending station. The  $v$  variables for initial trips and constraints (2.3a) are not essential for our model since  $W_i^{kp}$  is a known constant; however, we retain these variables and constraints for notational convenience when we later formulate the PSQ rules. Constraints (2.4a) and (2.4b) impose the capacity restrictions for taxis and other deadhead modes, respectively (for uncapacitated modes such as buses or passenger trains, we can omit these constraints). Observe that these two constraints link the crew assignment decisions across occupations. Constraints

(2.5) capture the PSQ rules that regulate feasible assignments; we develop detailed versions of these constraints in Section 2.4.2. Constraints (2.6) impose the requirement that we can assign an extra crew member only when none of the crew members in the primary or secondary queues (regular crews or away-from-home extra crews) are rested when a trip departs. The constraints enforce this condition by requiring that, if a trip  $j$  departing from station  $s$  is assigned an extra crew member homed at  $s$ , then no inbound trip  $i$  that could have connected to trip  $j$  should connect to a trip  $j'$  that departs later than  $j$ . Constraints (2.7) to (2.9) are the nonnegativity and integrality requirements. Note that we do not explicitly impose the integrality of the  $v$  variables since constraints (2.3a) to (2.3c) together with the integrality of the  $x$  variables imply that the  $v$  variables must be integer-valued.

#### 2.4.2 Constraints for PSQ Crew Dispatching Rules

We now explain how to express the crew dispatching rules in PSQ districts, represented by the set  $\mathbf{X}(RK, \beta)$  in formulation [CAP], as linear constraints. Throughout this section, for clarity, we drop the occupation index  $k$ , since all crew dispatching rules apply independently to each occupation. We say that crew member  $a$  has higher dispatching priority than another member  $b$  if, for any outbound trip  $j$  for which both crews are available (i.e., not already dispatched) and rested,  $a$  must be assigned first before assigning  $b$ . We next discuss how the rotation key and PSQ scheme determine the relative dispatching priorities for crews arriving at a station  $s$  on two different inbound trips  $i_1$  and  $i_2$ . Let crew- $i_1$  and crew- $i_2$  respectively denote the crew members arriving on these two trips. Suppose a crew member arriving on an inbound trip  $i_1$  has

higher rotation key order than one arriving on trip  $i_2$ , i.e., trip  $i_1$  arrives (departs) earlier than trip  $i_2$  for FIFO (FOFO) rotation key. We use the notation  $i_1 \prec i_2$  to denote this order. As a convention, for FIFO rotation key, we add a traveling crew member to the appropriate (primary or secondary) queue at the ending station when s/he arrives at this station; for FOFO, we add the crew member when s/he departs from the starting station. With this convention, since  $i_1 \prec i_2$ , crew- $i_1$  joins a queue at station  $s$  first, for either rotation key. We refer to crews based at station  $s$  as home crews and those based at the other station as away crews. Since crew- $i_1$  and crew- $i_2$  can each be a home or away crew member, we have four possible combinations of home bases for these two crew members: both are home crews, both are away crews, crew- $i_1$  is an away crew and crew- $i_2$  is a home crew, and crew- $i_1$  is a home crew while crew- $i_2$  is an away crew.

If crew- $i_1$  and crew- $i_2$  are from the same home base, then they join the same queue and their dispatching order is solely determined by the rotation key. If crew- $i_1$  is an away crew and crew- $i_2$  is a home crew, then crew- $i_1$  goes to the primary queue while crew- $i_2$  joins the secondary queue at a later time (and gets pushed to the primary queue later). In all these three situations, crew- $i_1$  has higher dispatching priority than and crew- $i_2$  since  $i_1 \prec i_2$ . In Section 2.4.2.1, we model these three rules as *Rotation Key constraints*.

Finally, if crew- $i_1$  is a home crew and crew- $i_2$  is an away crew, then crew- $i_1$  enters the secondary queue first, while crew- $i_2$  joins the primary queue at a later time. In this case, the relative dispatching priorities of these two crew members depend on whether or not crew- $i_1$  is pushed to the primary queue before crew- $i_2$  joins this queue. We discuss

this case in Section 2.4.2.2, and formulate it as *Push-Pull constraints* to capture the mechanism that push and pull home crew members from the secondary queue to the primary queue.

If two crew members arrive at station  $s$  simultaneously on the same inbound (deadhead) trip  $i$ , they have the same priority if they are from the same home base. These crews can then be assigned to outbound trips in arbitrary order, and thus no constraints are needed. If trip  $i$  is a deadheading trip that carries a mix of home and away crews, then the away crews are given higher priority. We model this condition as a special case of Push-Pull constraints with  $i_1 = i_2 = i$ .

#### 2.4.2.1 Rotation Key constraints

For any two trips  $i_1$  and  $i_2$  arriving at station  $s$ , with  $i_1 \prec i_2$ , let  $CR(i_1, i_2)$  denote the set of outbound trips from station  $s$  that are eligible to connect from both  $i_1$  and  $i_2$ , i.e.,  $CR(i_1, i_2) = \{j: \langle i_1, j \rangle \in IJ \text{ and } \langle i_2, j \rangle \in IJ\}$ . For any pair of outbound trips  $j$  and  $j'$  in  $CR(i_1, i_2)$ , with trip  $j$  departing from station  $s$  before trip  $j'$ , i.e.,  $td_j < td_{j'}$ , the rotation key rule implies that crew- $i_1$  has higher dispatching priority than crew- $i_2$  if they are from the same home base. Therefore, if crew- $i_2$  is assigned to trip  $j$ , then crew- $i_1$  is not allowed to take the later trip  $j'$ . Constraints (2.10) impose this rule:

$$\sum_{p \in P_s} x_{i_1 j'}^p + \sum_{p \in P_s} x_{i_2 j}^p \leq 1, \quad \forall i_1 \prec i_2, j, j' \in CR(i_1, i_2) \text{ and } td_j < td_{j'}, s \in \{A, B\}. \quad (2.10)$$

If crew- $i_1$  is an away crew and crew- $i_2$  is a home crew, then crew- $i_1$  also has higher priority in this case. Therefore, we may extend the rotation key rule such that crew- $i_1$  has higher priority if s/he is an away crew, regardless whether crew- $i_2$  is an away

crew or a home crew. The following constraints (2.10a) correspond to the situation when both crew- $i_1$  and crew- $i_2$  are home crews. Constraints (2.10b) are the extended constraints when crew- $i_1$  is an away crew and crew- $i_2$  is either a home or away crew.

$$\sum_{p \in P_s} x_{i_1 j'}^p + \sum_{p \in P_s} x_{i_2 j}^p \leq 1, \quad \forall i_1 \prec i_2, j, j' \in CR(i_1, i_2) \text{ and } td_j < td_{j'}, s = sa_{i_1} = sa_{i_2}, \quad (2.10a)$$

$$\sum_{p \in P_{s'}} x_{i_1 j'}^p + \sum_{p \in P} x_{i_2 j}^p \leq 1, \quad \forall i_1 \prec i_2, j, j' \in CR(i_1, i_2) \text{ and } td_j < td_{j'}, s' = sd_{i_1} = sd_{i_2}. \quad (2.10b)$$

Since at least one of the trips in every feasible connection must be a regular trip that requires only one crew member of each occupation type, none of the summations on the left-hand sides of these two constraints can exceed one. The constraints, therefore, specify that if the solution uses the connection  $\langle i_2, j \rangle$ , then it cannot use the connection  $\langle i_1, j' \rangle$ , and vice versa.

Note that constraints (2.10a) and (2.10b) are needed for every pair of inbound trips  $i_1$  and  $i_2$  having common successors (i.e.,  $CR(i_1, i_2) \neq \emptyset$ ), and not just for pairs of adjacent trips because each of these constraints corresponds to specific pools, and crews from these pools may not be assigned to consecutive trips (either because one of these trips use members from another pool or is an unused deadhead trip). Similarly, we need to consider all pairs of outbound trips  $j, j' \in CR(i_1, i_2)$  rather than just adjacent pairs. So, if  $n$  denotes the number of trips in the planning horizon, model [CAP] requires  $O(n^4)$  Rotation Key constraints. We later (in Section 2.5) discuss ways to both strengthen these constraints and reduce their number.

#### 2.4.2.2 Push-Pull constraints

When one or more home crews arrive on trip  $i_1$  and one or more away crews arrive on trip  $i_2$ , with  $i_1 \prec i_2$ , the relative dispatching priorities of these home and away crews depend on the queue position of the home crew at the time when the away crew joins the primary queue. Depending on the number of home crews arriving at station  $s$  on the trips between  $i_1$  and  $i_2$ , and the capacity of the secondary queue  $\beta_s$ , a home crew on trip  $i_1$  may be pushed to the primary queue by another home crew arriving on a trip before or after trip  $i_2$ . S/he could also be pushed by a home crew arriving on trip  $i_2$  if this trip is a deadhead trip carrying a mix of home and away crews. Based on our rule of breaking ties, a home crew arriving on trip  $i_1$  has higher priority than an away crew arriving on trip  $i_2$  if and only if the home crew is pushed to the primary queue before the away crews from trip  $i_2$  join this queue. For any given crew assignments, let  $v_i^H$  be the number of home crews assigned to trip  $i$ . Then,  $\Delta_{i_1 i_2}^1 = \min \left\{ (v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H - \beta_s)^+, v_{i_1}^H \right\}$  home crews from  $i_1$  have higher priority than the away crews from  $i_2$ . Therefore, if an away crew from trip  $i_2$  connects to an outbound trip  $j$ , then at least  $\Delta_{i_1 i_2}^1$  home crews from trip  $i_1$  should be assigned on or before trip  $j$ . Note that more than  $\Delta_{i_1 i_2}^1$  home crews can be assigned trips earlier than trip  $j$  if the away crews from trip  $i_2$  are not rested for those outbound trips (i.e., crew pull up). We can express this condition as (2.11a). To simplify the notation, let  $x_{ij}^H = \sum_{p \in P_s} x_{ij}^p$  and  $x_{ij}^A = \sum_{p \in P_{s'}} x_{ij}^p$  respectively represent number of home and away crews that transfer from an inbound trip  $i$  to an outbound trip



$j$ . Observe that  $x_{ij}^H, x_{ij}^A \in \{0,1\}$ .

$$\sum_{j': td_{j'} \leq td_j} x_{ij'}^H \geq \Delta_{i_1 i_2}^1 - M(1 - x_{i_2 j}^A), \quad \forall i_1 \prec i_2, j \in CR(i_1, i_2). \quad (2.11a)$$

Similarly,  $\Delta_{i_1 i_2}^2 = v_{i_1}^H - \Delta_{i_1 i_2}^1 = \min\left\{(\beta_s - \sum_{i_1 \prec i \prec i_2} v_i^H)^+, v_{i_1}^H\right\}$  home crews from  $i_1$  are pushed to the primary queue by trips on or after  $i_2$ , and thus have lower priority than the away crews from trip  $i_2$ . Therefore, if an away crew from trip  $i_2$  connects to an outbound trip  $j$ , we should assign  $\Delta_{i_1 i_2}^2$  home crews from trip  $i_1$  on or after trip  $j$ . However, if more than  $\Delta_{i_1 i_2}^1$  home crews from trip  $i_1$  are assigned to trips  $j'$  with  $td_{j'} \leq td_j$  because the away crews from trip  $i_2$  are not rested, then we should not enforce  $\Delta_{i_1 i_2}^2$  home crews from trip  $i_1$  to be assigned on or after trip  $j$ . This constraint is necessary only when one or more home crews from trip  $i_1$  connect to outbound trips  $j'$  with  $td_{j'} < td_j$  and  $j' \in CR(i_1, i_2)$ , that is,

$$\text{If } \sum_{j' \in CR(i_1, i_2), td_{j'} < td_j} x_{ij'}^H \geq 1, \text{ then } \sum_{j': td_{j'} \geq td_j} x_{ij'}^H \geq \Delta_{i_1 i_2}^2 - M(1 - x_{i_2 j}^A), \forall i_1 \prec i_2, j \in CR(i_1, i_2) \quad (2.11b)$$

Note that the expressions of  $\Delta_{i_1 i_2}^1$  and  $\Delta_{i_1 i_2}^2$  are nonlinear. If  $\sum_{i_1 \prec i \prec i_2} v_i^H < \beta_s$ ,

$$\Delta_{i_1 i_2}^1 = v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H - \beta_s; \text{ otherwise, } \Delta_{i_1 i_2}^1 = v_{i_1}^H. \quad \text{For } \Delta_{i_1 i_2}^2, \text{ it equals } v_{i_1}^H \text{ if}$$

$$v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H \leq \beta_s; \text{ otherwise, } \Delta_{i_1 i_2}^2 = \beta_s - \sum_{i_1 \prec i \prec i_2} v_i^H. \quad \text{To linearize constraints (2.11a) and}$$

(2.11b), we introduce two indicators for each pair of trips  $i_1$  and  $i_2$  with  $i_1 \prec i_2$ :

$$u_{i_1 i_2}^1 = \begin{cases} 0, & \text{if } \sum_{i_1 \prec i \prec i_2} v_i^H < \beta_s \\ 1, & \text{if } \sum_{i_1 \prec i \prec i_2} v_i^H \geq \beta_s \end{cases}, \text{ and}$$

$$u_{i_1 i_2}^2 = \begin{cases} 0, & \text{if } v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H \leq \beta_s \\ 1, & \text{if } v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H > \beta_s \end{cases}.$$

Further, we need one more indicator for each triplet  $i_1$ ,  $i_2$ , and  $j$ , with  $i_1 \prec i_2$  and  $j \in CR(i_1, i_2)$  to capture the IF-condition in (2.11b):

$$u_{i_1 i_2 j}^3 = 1, \text{ if } \sum_{j' \in CR(i_1, i_2), td_{j'} < td_j} x_{i_1 j'}^H \geq 1.$$

Appendix A presents the forcing constraints for the three indicator variables. Constraints (2.12a) to (2.12d) are the linearized Push-Pull constraints, incorporating the indicators and appropriate expression of  $\Delta_{i_1 i_2}^1$  and  $\Delta_{i_1 i_2}^2$ .

$$\sum_{j': td_{j'} \leq td_j} x_{i_1 j'}^H \geq \left( v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H - \beta_s \right) - M(1 - x_{i_2 j}^A) - M u_{i_1 i_2}^1, \quad \forall i_1 \prec i_2, j \in CR(i_1, i_2), \quad (2.12a)$$

$$\sum_{j': td_{j'} \leq td_j} x_{i_1 j'}^H \geq v_{i_1}^H - M(1 - x_{i_2 j}^A) - M(1 - u_{i_1 i_2}^1), \quad \forall i_1 \prec i_2, j \in CR(i_1, i_2), \quad (2.12b)$$

$$\sum_{j': td_{j'} \geq td_j} x_{i_1 j'}^H \geq v_{i_1}^H - M(1 - x_{i_2 j}^A) - M u_{i_1 i_2}^2 - M(1 - u_{i_1 i_2 j}^3), \quad \forall i_1 \prec i_2, j \in CR(i_1, i_2), \quad (2.12c)$$

$$\sum_{j': td_{j'} \geq td_j} x_{i_1 j'}^H \geq \left( \beta_s - \sum_{i_1 \prec e \prec i_2} v_e^H \right) - M(1 - x_{i_2 j}^A) - M(1 - u_{i_1 i_2}^2) - M(1 - u_{i_1 i_2 j}^3), \quad (2.12d)$$

$$\forall i_1 \prec i_2, j \in CR(i_1, i_2).$$

Constraints (2.12b) state that if  $x_{i_2 j}^A = 1$ , then all home crews from trip  $i_1$  must connect to trips on or before trip  $j$ . Equivalently, we can say that none of the home

crews from trip  $i_1$  should connect to trips after trip  $j$ , i.e.,  $\sum_{j': td_{j'} > td_j} x_{i_1 j'}^H = 0$ . Therefore, we

can re-write constraints (2.12b) as follows:

$$\sum_{j': td_{j'} > td_j} x_{i_1 j'}^H \leq M(1 - x_{i_2 j}^A) + M(1 - u_{i_1 i_2}^1), \quad \forall i_1 \prec i_2, j \in CR(i_1, i_2). \quad (2.12b^*)$$

In the same spirit, constraints (2.12c) enforce that all home crews from trip  $i_1$  are assigned on or after trip  $j$ . Alternatively, we can enforce that no away crews from trip  $i_2$  are assigned to trips after  $j$  if any home crew from trip  $i_1$  connects to  $j$ , that is,

$$\sum_{j': td_{j'} > td_j} x_{i_2 j'}^A \leq M(1 - x_{i_1 j}^H) + M u_{i_1 i_2}^2, \quad \forall i_1 \prec i_2, j \in CR(i_1, i_2). \quad (2.12c^*)$$

Next, we discuss three special cases for which we can simplify the Push-Pull constraints. The first special case handles the situation where we have both home and away crews arriving on the same trip, i.e.,  $i_1 = i_2$ . Since in this case the away crews always have higher priority, we only need to impose constraints (2.12c\*) with  $u_{i_1 i_2}^2$  fixed at 0. The second special case applies when all trips between trips  $i_1$  (inclusive) and  $i_2$  (exclusive) are initial trips (i.e., trips that have already completed or are in progress at the start of the horizon). In this case, we know the actual number of home crews joining the secondary queue at station  $s$  between trips  $i_1$  and  $i_2$  (based on the prior crew assignments to the initial trips). So,  $\Delta_{i_1 i_2}^1$  and  $\Delta_{i_1 i_2}^2$  are known constants, and we do not need any indicator variables to linearize them. The last special case applies when the secondary queue has a zero capacity (i.e.,  $\beta_s = 0$ ). In this case, all home and away crews join the primary queue directly, and their dispatching priorities are fully determined by the rotation key. Then, the Rotation Key constraints (2.10a) and (2.10b), and the Push-Pull

constraints (2.12a), (2.12b\*), (2.12c\*) and (2.12d) reduce to the following expanded Rotation Key constraints that apply across all crew pools:

$$\sum_{p \in P} x_{i_1 j'}^p + \sum_{p \in P} x_{i_2 j}^p \leq 1, \quad \forall i_1 \prec i_2, j, j' \in CR(i_1, i_2) \text{ and } td_j < td_{j'}. \quad (2.13)$$

Finally, we note that the full model requires  $O(n^3)$  Push-Pull constraints and  $O(n^4)$  Rotation Key constraints, where  $n$  is the number of trips. For instance, a problem with  $n = 190$  trips requires over 581,000 such constraints. Since these constraints are so numerous, our implementation does not include all Rotation Key and Push-Pull constraints in the initial model. Rather, we only include a small subset (discussed in Section 2.7.1) of these constraints in the initial model, and let the solver add the remaining constraints as needed during the solution process.

## 2.5 MODEL ENHANCEMENTS

To reduce the computational time for solving the crew assignment problem, we develop: (i) stronger constraints and valid inequalities to tighten the formulation so as to increase the lower bound provided by the linear programming (LP) relaxation, and (ii) methods to reduce the problem size by eliminating some connection variables and constraints. As our later computational results (in Section 2.7) show, these techniques are very effective for accelerating the solution procedure.

### 2.5.1 Deadhead Capacity Cuts

Constraints (4a) in formulation [CAP] serve both to impose the capacity  $U_i$  for each taxi and to relate the taxi selection variable  $y_i$  to the assignment decisions to this deadhead trip. Since  $U_i$  is greater than one, this constraint can yield a weak LP

relaxation, i.e., the  $y_i$  variable can take a value as low as  $1/U_i$  (when one crew member is assigned to this trip), thus permitting the LP solution to incur only a fraction of the taxi's fixed cost. To strengthen these constraints, we add the following set of disaggregated forcing constraints:

$$\sum_{p \in P} x_{ij}^{kp} \leq y_j, \quad \forall j \in DT, \langle i, j \rangle \in IJ, k \in K. \quad (2.14)$$

These forcing constraints exploit the property that the sum of the assignment variables in the left-hand side can never exceed one (since trip  $i$  must be a regular trip that requires no more than one crew member for each occupation). Analogous disaggregate forcing constraints have proved very useful for tightening the LP relaxation of problems in other contexts, particularly in the facility location and network design literature (e.g., Balakrishnan et al., 1989).

### 2.5.2 Stronger Rotation Key Constraints

Recall that the Rotation Key constraints (2.10a) and (2.10b) impose dispatching priorities by requiring the sum of two mutually conflicting assignments (of crew members from inbounds trips  $i_1$  and  $i_2$ ) to be less than or equal to one. We can strengthen the formulation by adding more mutually conflicting  $x$  variables on the left hand side of the inequality while keeping the right hand side at one. The key property we exploit is that  $\sum_{j: \langle i, j \rangle \in IJ} \sum_{p \in P} x_{ij}^p$  must not exceed one if trip  $i$  can carry at most one crew member (unit capacity); similarly,  $\sum_{i: \langle i, j \rangle \in IJ} \sum_{p \in P} x_{ij}^p \leq 1$  if trip  $j$  has unit capacity.

Let crew- $i_1$  and crew- $i_2$  be two home crew members arriving on trips  $i_1$  and  $i_2$  with  $i_1 \prec i_2$ . The Rotation Key constraints (2.10a) state that if crew- $i_2$  is assigned to

outbound trip  $j$  (for which crew- $i_1$  is also rested), then crew- $i_1$  is not allowed to take a later trip  $j'$ . We can strengthen this condition as follows: if any crew member from trip  $i_2$  connects to a common outbound trip  $j$ , then crew- $i_1$  should not take *any* trip that departs later than  $j$ , that is,  $\left(\sum_{p \in P_s} x_{i_2 j}^p = 1\right) \Rightarrow \left(\sum_{j': td_{j'} > td_j} \sum_{p \in P_s} x_{i_1 j'}^p = 0\right)$ . Note that the summation in the second expression is at most one if trip  $i_1$  has unit capacity. Therefore, we can enforce the sum of these two expressions to be less than or equal to one. Similarly, if crew- $i_1$  is assigned to trip  $j$ , then crew- $i_2$  cannot take *any* outbound trip  $j'$  that departs before  $j$ , unless crew- $i_1$  is not rested for  $j'$ . If trip  $i_2$  has unit capacity, then  $\sum_{j' \in CR(i_1, i_2), td_{j'} < td_j} \sum_{p \in P_s} x_{i_2 j'}^p \leq 1$ . Further, if trip  $j$  is a unit capacity trip, then at most one of crew- $i_1$  and crew- $i_2$  can connect to trip  $j$ . We can use these observations, illustrated in Figure 2.3, to strengthen the Rotation Key constraints whenever the respective unit capacity condition holds.

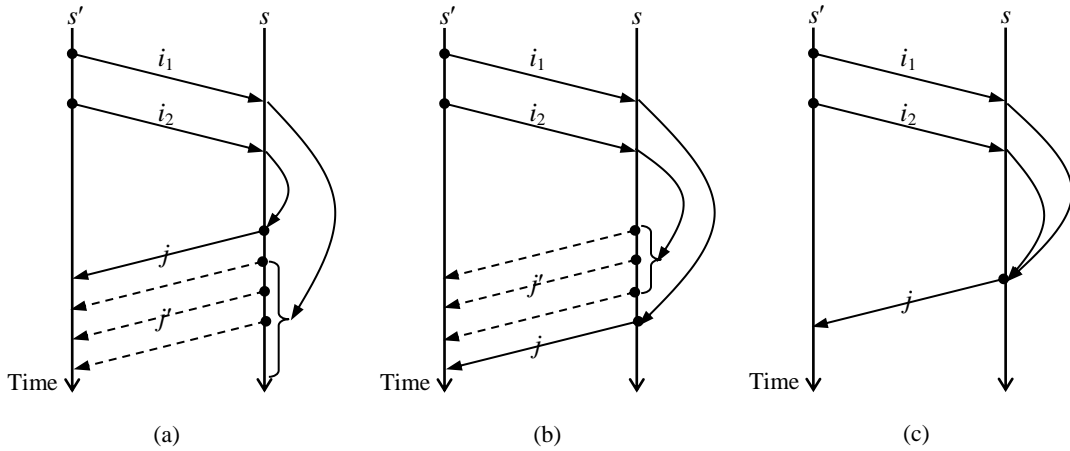


Figure 2.3 Stronger Rotation Key constraints with unit capacity trip  
(a) trip  $i_1$  has unit capacity (b) trip  $i_2$  has unit capacity (c) trip  $j$  has unit capacity

Note that when trip  $i_1$  and/or trip  $i_2$  has unit capacity, the strengthened version of the Rotation Key constraints has fewer constraints since, for every pair of inbound trips  $i_1$  and  $i_2$ , we require only one constraint for every outbound trip  $j$  (whereas for the basic version (2.10a), we need one constraint for every pair of outbound trips  $j$  and  $j'$ ). Table 2.1 summarizes the strengthened Rotation Key constraints corresponding to (2.10a), for all possible combinations of inbound and outbound trip capacities. The same strengthening scheme applies to (2.10b) since it has the same structure as (2.10a) except that we sum variables for different crew pools in these two constraints.

Trip capacity			Rotation Key constraints for trip pair $i_1 \prec i_2$ arriving at station $s$
Trip $i_1$	Trip $i_2$	Trip $j$	
$\geq 2$	$\geq 2$	1	$\sum_{p \in P_s} x_{i_1 j'}^p + \sum_{p \in P_s} x_{i_2 j}^p \leq 1, \forall j, j' \in CR(i_1, i_2) \text{ and } td_j < td_{j'}$
1	$\geq 2$	1	$\sum_{j': td_{j'} \geq td_j} \sum_{p \in P_s} x_{i_1 j'}^p + \sum_{p \in P_s} x_{i_2 j}^p \leq 1, \forall j \in CR(i_1, i_2)$
$\geq 2$	1	1	$\sum_{p \in P_s} x_{i_1 j}^p + \sum_{j' \in CR(i_1, i_2), td_{j'} \leq td_j} \sum_{p \in P_s} x_{i_2 j'}^p \leq 1, \forall j \in CR(i_1, i_2)$
1	1	$\geq 2$	$\sum_{j': td_{j'} > td_j} \sum_{p \in P_s} x_{i_1 j'}^p + \sum_{j' \in CR(i_1, i_2), td_{j'} \leq td_j} \sum_{p \in P_s} x_{i_2 j'}^p \leq 1, \forall j \in CR(i_1, i_2)$
1	1	1	$\sum_{j': td_{j'} \geq td_j} \sum_{p \in P_s} x_{i_1 j'}^p + \sum_{j' \in CR(i_1, i_2), td_{j'} \leq td_j} \sum_{p \in P_s} x_{i_2 j'}^p \leq 1, \forall j \in CR(i_1, i_2)$

Table 2.1 Variations of Rotation Key constraints based on constraints (2.10a)

### 2.5.3 Problem Reduction: Latest Connection

As another enhancement, we exploit opportunities to reduce the problem size based on the properties of solutions that satisfy the PSQ rules. The PSQ scheme determines crew-to-trip assignments using a one-to-one matching procedure from the crew queues to the outbound trip list. At each station, we can construct the initial crew

queues based on the initial trips (already completed or enroute) arriving at that station. By exploiting the restriction that we cannot use extra crews if any initial crew member is rested and available for a trip and by ignoring deadhead trips, the *latest-connection identification* procedure, shown in Figure 2.4 and discussed next, finds the latest possible outbound connection  $j^*(i)$  for some or all initial trips  $i$ . We can then eliminate all variables corresponding to connections from trip  $i$  to trips departing later than  $j^*(i)$ . Eliminating these connection variables also reduces the number of trips in the set  $CR(i, i')$  when  $i$  and/or  $i'$  is an initial trip, thereby significantly reducing the number of Rotation Key constraints and Push-Pull constraints. We note that this method not only reduces the size of the problem, but can also strengthen the LP relaxation and raise the initial lower bound (as shown by our computational results in Section 2.7).

To identify the latest connections for initial trips arriving at station  $s$ , we first construct the initial crew queues (primary and secondary) at that station based on the information of these inbound initial trips. If the rotation key is FIFO, we delete from the initial queues those crews who reach station  $s$  after the first (regular or deadhead) future trip from the other stations arrives at station  $s$ . (For the FOFO rotation key, since all initial crews departed before the first future inbound trip departs, we retain all of them in the queues.) After arranging all future *regular* trips leaving station  $s$  in departure time order, we identify the latest outbound connection for the crews in the initial queues in two steps. Let  $t_1$  denote the time at which the first future trip arrives at station  $s$ . Since the crew(s) arriving on this trip are not adequately rested until time  $t^* = t_1 + \text{minimum rest time}$ , we first sequentially assign crews from the initial queues (primary or



secondary) to regular trips departing before  $t^*$  (or use extra crews based at station  $s$ , if no initial crew in the queues is rested) in departure time order. The outbound trip to which this procedure assigns each of these crew members defines the latest connection for the initial trip on which the crew member arrived. This property holds because we have ignored outbound deadhead trips during the matching process (with intermediate deadhead trips, the solution may dispatch some of these crew members earlier, but never later). In the second step, we determine the latest outbound connections for any remaining initial crews in the primary queue. Note that the second step only applies to the primary queue, and not the secondary queue, because crew(s) arriving on the first future trip may be away crew(s) who will join the primary queue at  $s$  and so have higher dispatching priority than the remaining initial crews in the secondary queue.

Next, we establish the validity of the latest-connection identification procedure.

**Claim 2.1:**

The outbound trip  $j^*(i)$  found by the latest-connection identification procedure is the latest feasible outbound connection for initial trip  $i$ , assuming that usage of extra crews is not allowed when one or more initial crew members are available and rested.

**Proof:** Let  $j^*(i)$  be the outbound trip that the procedure assigns to a crew member arriving on an initial trip  $i$ . Suppose the problem has a feasible solution that assigns the crew from trip  $i$  to a trip  $j'$  that departs later than  $j^*(i)$ . Then,  $j^*(i)$  must be assigned to a crew member from another initial trip  $i'$  since the crew from trip  $i$  is available and rested when  $j^*(i)$  departs and we are not permitted to use an extra crew member in this case. Moreover, the crew on trip  $i'$  must have lower priority than the person on trip  $i$  because the procedure guarantees that the crew on trip  $i$  has the highest priority among all

unassigned crews that are rested for  $j^*(i)$ . Therefore, the solution with connections  $\langle i', j \rangle$  and  $\langle i, j' \rangle$  violates the PSQ dispatching rules, contradicting with the feasibility of the given solution. ♦

<b>Initialization:</b>	
Let $PQ\_init(s)$ = (partial) initial primary queue at station $s$ , $SQ\_init(s)$ = (partial) initial secondary queue at station $s$ , $R_s$ = list of regular trips departing from station $s$ , ordered by departure time, $t^*$ = projected rested time of crews arriving on the first future trip inbound $s$ .	
<b>Step 1:</b> Assign outbound trips departing before $t^*$ :	<b>Step 2:</b> Assign remaining crews in $PQ\_init(s)$ :
Let $j := \text{first trip in } R_s$ <b>While</b> $td_j < t^*$ <b>do</b> Let $i := \text{first crew member in } PQ\_init(s) \text{ that is rested for } j$ <b>If</b> $i = \text{null}$ Let $i := \text{first crew member in } SQ\_init(s) \text{ that is rested for } j$ <b>End If</b> <b>If</b> $i = \text{null}$ Assign an extra crew member based at $s$ to trip $j$ <b>Else</b> $j^*(i) := j$ Delete connections $\langle i, j' \rangle, \forall j' : td_{j'} > td_j$ Delete trip $i$ from $PQ\_init(s)$ or $SQ\_init(s)$ <b>End If</b> $j := \text{next trip in } R_s$ <b>End</b>	<b>While</b> $PQ\_init(s) \neq \emptyset$ <b>do</b> Let $i = \text{first crew member in } PQ\_init(s)$ Let $j = \text{first unassigned trip in } R_s \text{ for which } i \text{ is rested}$ $j^*(i) := j$ Delete connection $\langle i, j' \rangle, \forall j' : td_{j'} > td_j$ Delete trip $i$ from $PQ\_init(s)$ <b>End</b>

Figure 2.4 The latest-connection identification procedure

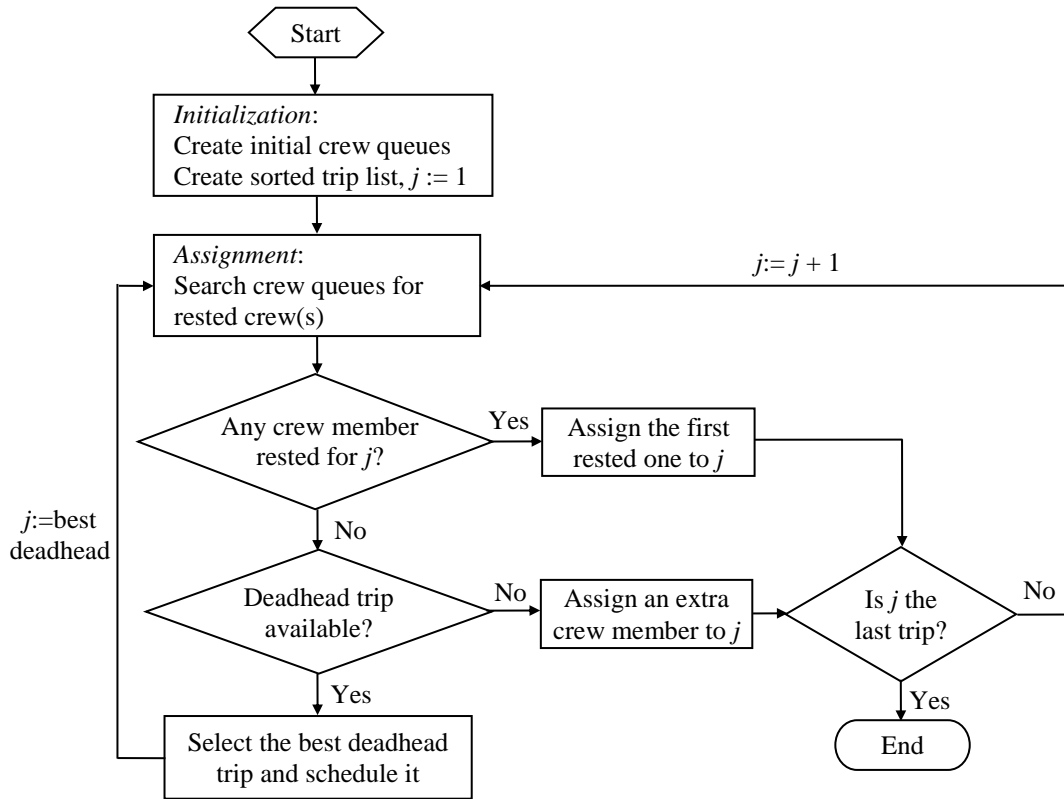


Figure 2.5 Flow chart of the crew assignment procedure

## 2.6 CREW ASSIGNMENT HEURISTIC

We next propose a heuristic algorithm that can provide high quality solutions very quickly. These solutions can be used to warm-start and accelerate the branch-and-bound procedure, and can also serve as suggested crew assignments to crew dispatchers. The heuristic algorithm consists of two stages. The first stage constructs an initial feasible solution by adding deadhead trips or using extra crews to sequentially resolve crew shortages on regular trips. The second stage then applies several local improvement steps to reduce total cost. In the first stage (and when evaluating the local improvement moves in the second stage), the *crew assignment procedure* assigns crews

to trips by applying the PSQ scheme. Figure 2.5 shows a flow chart of this procedure. Note that initially each candidate deadhead trip has a crew requirement of zero. If, in a later step or stage, we decide to assign one or more crew members to a deadhead trip, we set the minimum required crews on this trip equal to the number assigned and treat this trip as a “regular” trip whose minimum crew requirement must be fulfilled.

### ***Crew assignment procedure***

#### *Step 1: Initialization*

- Construct the initial crew queues at both stations based on initial trip information.
- List all (regular and candidate deadhead) trips departing from either station in increasing order of departure time. Index the trips from 1 to  $n$  in this order.
- Let  $j$  be the index of the current trip to be staffed. Initialize  $j := 1$ .

#### *Step 2: Assignment*

- For trip  $j$  and for each occupation,
- For every required crew member on this trip, search the primary queue and then the secondary queue at the departure station of trip  $j$  for the first rested crew member in the queue, and assign this crew member to trip  $j$ .
  - Remove the assigned crew from the current queue, and add him/her to the appropriate (primary or secondary) queue at the destination station of trip  $j$ .
  - If there is no crew in either the primary or secondary queue rested for trip  $j$ , go to Step 3.
  - Stop if  $j = n$ . Otherwise, set  $j := j + 1$ , and repeat Step 2.

#### *Step 3: Resolving crew deficit*

- Let  $D(j)$  be the set of candidate deadhead trips that can connect to trip  $j$ .
- If  $D(j) = \emptyset$ , assign an extra crew member to trip  $j$  and add this crew member to the primary queue at the arrival station of  $j$ . Then go to Step 2.
- If  $D(j) \neq \emptyset$ , select the least cost inbound deadhead trip  $i \in D(j)$  using the *deadhead cost evaluation* procedure described below, add a crew requirement of one to trip  $i$ , set  $j := i$ , and return to Step 2.

### ***Deadhead cost evaluation procedure:***

Given a trip  $j$  to be staffed by a deadheaded crew member and an inbound deadhead trip  $h$  to be evaluated, tentatively add a crew requirement of one to trip  $h$ .

Perform the crew assignment routine described in Steps 1 and 2 (with non-zero crew requirements for each previously chosen deadhead trip), but using extra crews to resolve any remaining crew shortages. Calculate the total cost of the solution using the proposed additional deadhead on trip  $h$ , including assignment, layover, deadheading, and extra crew costs.

The first stage considers trips one at a time and uses myopic rules to decide the transport modes for deadheading crews. For instance, the procedure may choose to deadhead a crew member by train or on public transport, but may later add a taxi (to deadhead another crew member) that could have also transported this crew member and saved on his/her transportation fare and/or layover cost. Further, Stage 1 deploys an extra crew member only if crews from the other station cannot be deadheaded in time. And, it does not consider deadheading away crews who have long layovers (and therefore incur high layover costs) back to their home stations. To reduce the cost of the solution generated by the first stage, Stage 2 applies various local improvement steps to exploring additional deadheading options. These steps include:

- Reassigning deadheaded crews to take advantage of currently scheduled taxi trips;
- Rescheduling (e.g., advancing) the deployment of extra crews used by the current solution so as to defer or eliminate deadhead trips that serve to cover crew shortages;
- Reducing heldaway costs by deadheading additional crews. Note that deadheading one crew member may reduce layover time for several other (subsequent) crew members since the assignments of crews with lower priorities than the one who is deadheaded are moved forward by one trip.

For each of these moves or interchanges, we reapply the Stage 1 crew assignment procedure after appropriately adjusting the costs or required number of crew members for select deadhead trips. We apply these steps iteratively, updating the current best solution whenever the cost decreases. The Stage 2 procedure terminates when it does not yield further improvement.

## 2.7 COMPUTATIONAL RESULTS

To assess the effectiveness and robustness of our model enhancements and solution techniques, we applied them to 140 real-life problem instances, each with a planning horizon of 36 hours, sampled from 14 PSQ crew districts of a U.S. freight railway company. For each problem instance, we generate the candidate deadhead trips as follows:

- For each regular trip with locomotive seats that can be used for deadheading, we create a train deadhead trip with the same departure and arrival times (plus a small time perturbation to ensure distinct times and avoid ties in crew assignment) and stations as the regular trip;
- Let  $T_r$  be the minimum required rest time. For each regular trip, we create an inbound taxi trip that reaches the station exactly  $T_r$  time units before the regular trip's departure, and an outbound taxi trip that departs  $T_r$  time units after the regular trip arrives. Further, for each initial trip, we only create a corresponding outbound taxi trip. If crew dispatchers prohibit using taxis during certain specified "curfew" periods, we omit such taxis, but add extra taxis that start (end) immediately after (before) a curfew period ends (starts).

We implemented the optimization model (using CPLEX 12.3 as the solver) and the heuristic procedure in Java 1.5, on a computer with two 2.67 GHz Intel Core i5 M560 processors and 4 GB RAM.

Next, in Section 2.7.1, we demonstrate the effectiveness of the enhancements and methods developed in Sections 2.5 and 2.6, comparing computational results for selected problem instances with and without these enhancements. Section 2.7.2 summarizes the overall results for the 140 problems. Section 2.7.3 discusses the impacts of problem parameters, namely, the capacities of secondary queues and cost parameters, on the structure of optimal solutions and the performance of the solution methods.

### **2.7.1 Performance Analysis**

This section examines the effectiveness of the strong Rotation Key constraints (Section 2.5.2), latest-connection problem reduction method (Section 2.5.3), and the heuristic procedure (Section 2.6). We focus on (i) the reduction in the size of the model formulation, and (ii) improvement in the LP lower bound due to the stronger constraints and problem reduction, and then compare the computational performance with and without the enhancements. For this comparison, we consider four models. The *Base* model includes all constraints of model [CAP] and deadhead capacity cuts (constraints (2.13)). This model does not incorporate the stronger version of Rotation Key constraints or any problem reduction. Model *S* uses the stronger (and fewer) Rotation Key constraints (as shown in Table 2.1). Model *S+R* builds upon Model *S* by incorporating the latest-connection problem reduction method and the stronger Rotation Key constraints for the connections that are not eliminated. Our *Final* model includes

all of the enhancements of Model  $S+R$  plus the following two features: (i) we apply the heuristic method first, and provide the heuristic solution as the initial feasible solution for the branch-and-bound procedure (using the warm-start feature in CPLEX); and, (ii) we include only a subset of the Rotation Key and Push-Pull constraints in the initial model (as discussed below), and provide the remaining constraints as *lazy constraints* in CPLEX (CPLEX Studio V12.3, 2011). At intermediate stages of the branch-and-bound procedure, if CPLEX generates an integral solution that violates one or more of these lazy constraints, it dynamically adds them to the active model.

To conduct the performance comparisons for these four models, we focus on seven difficult problem instances from three different crew districts out of the 140 instances. Table 2.2 summarizes the dimensions of the seven problem instances and the size of the *Base* model formulation. In this sample of problem instances, the number of regular trips varies from 39 to 71, and the number of regular crew members (per occupation) ranges between 27 and 56. The total number of binary variables for these problem instances, including connection and taxi selection variables as well as the binary indicator variables used to linearize the Push-Pull constraints, ranges from 19,700 to 42,000, and the number of crew dispatching constraints (Rotation Key and Push-Pull constraints) varies from 172,000 to 581,000. Observe that the rotation key rule requires the most number of constraints since we need to consider all inbound trip pairs  $i_1 \prec i_2$  and all outbound trip pairs  $j, j'$  in  $CR(i_1, i_2)$ .



ID	Problem size				Model size of the <i>Base</i> model			
	# of engineers	# of conductors	# of regular trips	# of deadhead trips	# of x variables	# of other binary variables <sup>1</sup>	# of RK constraints	# of Push-Pull constraints
<b>J1</b>	28	27	39	48	4,213	15,657	116,486	56,089
<b>J2</b>	29	28	44	48	4,326	15,394	107,478	47,059
<b>L1</b>	40	46	65	63	8,304	33,681	437,409	142,694
<b>L2</b>	38	40	55	50	6,154	16,059	182,377	65,544
<b>N1</b>	56	-- <sup>2</sup>	63	62	4,750	15,919	216,585	78,622
<b>N2</b>	56	--	64	79	5,224	18,977	218,488	83,109
<b>N3</b>	56	--	71	70	5,621	21,972	316,815	100,492

<sup>1</sup> Taxi selection variables and binary indicator variables

<sup>2</sup> Only engineer is considered for District N

Table 2.2 Problem size of selected instances and model size of the Base model

ID	Model <i>S</i>		Model <i>S+R</i>			
	% reduction <sup>5</sup> in RK constraints	% increase <sup>6</sup> in LP bound	% reduction in x variables	% reduction in RK constraints	% reduction in Push-Pull constraints	% increase in LP bound
<b>J1</b>	62.37	0.00	20.34	64.14	21.64	0.00
<b>J2</b>	71.69	0.20	29.10	74.40	25.98	0.38
<b>L1</b>	70.39	6.38	34.67	72.55	39.29	13.88
<b>L2</b>	83.19	4.66	43.65	86.26	46.01	9.83
<b>N1</b>	71.47	0.12	28.51	73.09	32.33	0.22
<b>N2</b>	79.83	1.08	30.15	81.07	34.80	1.68
<b>N3</b>	66.64	1.52	25.55	67.57	19.19	2.29

<sup>3</sup> Model *S*: use stronger Rotation Key constraints

<sup>4</sup> Model *S+R*: use stronger Rotation Key constraints and apply latest-connection reduction

<sup>5</sup> Percentage reduction relative to the size of the *Base* model

<sup>6</sup> Percentage increase relative to the LP value of the *Base* model

Table 2.3 Model size reduction and LP bound improvement: Models *S*<sup>3</sup> and *S+R*<sup>4</sup>

Table 2.3 reports the reduction in model size and improvement in LP bound by using the stronger Rotation Key constraints and applying the latest-connection reduction method. Compared to the *Base* model, Model *S* has 62% to 83% fewer Rotation Key constraints, but the same number of variables and Push-Pull constraints (since they are not affected by the strengthening of the Rotation Key constraints). For Model *S+R*, the latest-connection reduction, combined with the stronger Rotation Key constraints, eliminated 20% to 44% of the connection variables, 64% to 86% of the Rotation Key constraints and 19% to 46% of the Push-Pull constraints. Further, both Model *S* and Model *S+R* provide tighter LP lower bounds than the *Base* model (the increase in lower bounds ranges from 0 to 13.88%). The improvement in LP bound is higher for Model *S+R* than for Model *S*, indicating that the latest-connection reduction method also contributes to strengthening the model formulation. Our computational results showed that the improvement of LP bounds may vary significantly, depending on the initial distribution of crew members, capacities of the secondary queues, and the schedule of trains to be assigned of each problem instances. Generally, the LP solutions of the *Base* model are more fractional, and tend to send the away crews back home earlier and keep crews at their home station longer. Further, if there are extra crews that are initially at an away station, the *Base* LP solution may try to send out regular crews before sending the extra crews because regular crews can be assigned to later trips departing from the other station once they get rested. The stronger Rotation Key constraints and latest-connection restriction invalidate such fractional solutions, forcing the corresponding LP solutions to be less fractional and better observe the PSQ rules.

ID	<i>Base model</i>		<i>Model S+R</i>		<i>Final model</i> <sup>7</sup>		Heuristic gap <sup>9</sup> (%)
	Gap <sup>8</sup> (%)	CPU time (sec)	Gap (%)	CPU time (sec)	Gap (%)	CPU time (sec)	
<b>J1</b>	<i>NFS</i> <sup>10</sup>	3,358	0	217	0	17	0
<b>J2</b>	0	3,322	0	30	0	18	0.23
<b>L1</b>	<i>NFS</i>	3,245	0	271	0	56	0.75
<b>L2</b>	<i>NFS</i>	3,428	0	81	0	12	0
<b>N1</b>	2.91	3,457	0	219	0	91	0.13
<b>N2</b>	7.35	3,184	0	47	0	12	0
<b>N3</b>	<i>NFS</i>	3,008	0	1,227	0	70	0

<sup>7</sup> Apply warm start and lazy constraints upon Model *S+R*

<sup>8</sup> Percentage gap of the best known solution relative to the best lower bound

<sup>9</sup> Percentage gap of the heuristic solution relative to the optimal value

<sup>10</sup> No feasible solution

Table 2.4 Computational performance comparison: Base, Model *S+R*, and Final

Table 2.4 compares the computational performance of the *Base* model, Model *S+R*, and the *Final* Model. For the *Base* model, we specified a time limit of one hour for terminating the branch-and-bound procedure. Within this time, the *Base* model failed to obtain even a feasible solution for four out of the seven instances (J1, L1, L2, and N3). For two other problems (N1 and N2), the procedure terminated with gaps of 2.91% and 7.35% (the relative gap between the value of the best known solution and the final lower bound expressed as a percentage of the lower bound), respectively. Instance J2 was solved optimally using almost one hour. Model *S+R* solved all seven instances optimally, using only a few to 20 minutes. In the *Final* model, the initial model only includes the constraints for the special case of the Push-Pull constraints when all trips between  $i_1$  and  $i_2$  are initial trips (see Section 2.4.2.2); we designate the remaining (over 94% of the total) Push-Pull constraints as lazy constraints in CPLEX. Further, for trips

$i_1$  and  $i_2$  that depart several hours apart, we treat the corresponding Rotation Key constraints as lazy constraints since crews arriving on these two trips are less likely to compete for the same outbound trip and violate the rotation key rule. Finally, we use the heuristic solutions to warm start the branch-and-bound procedure. The heuristic solutions are near optimal, and have gaps less than 1% (the relative gap between the costs of the heuristic solution and the optimal solution expressed as a percentage of the optimal value). CPLEX added less than 2% of the lazy constraints to the active model. The *Final* model solved all these instances within 2 minutes.

The comparison of these four models shows that our solution approach is very effective, and the computational performance of the *Final* model is very promising. We believe the following factors all contribute to the good performance of the *Final* model:

- The strengthening of the Rotation Key constraints and the latest-connection reduction significantly reduce the model size (by about 30% in the number of connection variables and Push-Pull constraints and about 70% in the number of the Rotation Key constraints);
- Treating many of the Rotation Key and Push-Pull constraints as lazy constraints dramatically reduces the size of the initial model and makes it much easier to solve;
- The stronger Rotation Key constraints and the latest-connection reduction also help to tighten the LP lower bound (by up to 14%);
- The high quality heuristic solution provides a tight upper bound and accelerates the branch-and-bound process.

### 2.7.2 Computational Results by Crew District

To conduct a more comprehensive set of computational tests and verify the effectiveness of our methods, we applied our *Final* model to 10 problem instances (corresponding to the crew assignment problem facing planners at 10 different dates and times) from each of 14 different PSQ crew districts. Table 2.5 presents statistics on the problem and model sizes, quality of the heuristic solution, and the computational time for creating and solving the CPLEX model. All statistics reported in this table (except the last column showing the maximum CPU time over all instances) are averaged over the 10 instances for each district. For each problem instance, we first applied the heuristic algorithm, and provided the heuristic solution to CPLEX as an initial solution. The heuristic procedure required only a few seconds of CPU time for each problem, and the solution was optimal for 125 out of the 140 instances (89%). Further, among the 15 instances with positive heuristic gaps, only 3 instances had a gap over 1%. CPLEX was able to solve all problems to optimality within 2 minutes, and required less than 30 seconds on average to solve the *Final* model.

The following observations explain the variation in results across districts and instances:

- Districts A to E are small districts with fewer than 15 regular trips in a 36-hour period, and are therefore easy to solve;
- District G is easy to solve despite its high traffic volume because this district has zero-capacity secondary queues, and so the Rotation Key constraints and Push-Pull constraints reduce to the simplified constraints (2.13);

- Districts F, H and K are not highly capacity constrained in terms of crew availability. They are, therefore, easier to solve compared to other districts with similar traffic volumes (e.g., district I);
- The computational time for various problem instances from the same district can vary significantly. For instance, the maximum CPU time for district L was 56 seconds, whereas the average time was only 19 seconds. Possible factors contributing to this difference include initial crew availability at the two stations, traffic volume during the planning horizon, traffic distribution over time, and the imbalance of traffic in two directions.

District	Problem size				Model size of the <i>Final</i> model			Heuristic gap (%)	Soln time	
	# of eng	# of cond	# of reg trips	# of dh trips	# of $x$ var	# of RK constr	# of Push-Pull constr		Avg Time (sec)	Max time (sec)
<b>A</b>	5	5	5	13	161	62	175	0	0.05	0.08
<b>B</b>	9	9	11	15	327	455	992	0	0.11	0.16
<b>C</b>	9	10	8	18	306	252	888	0	0.11	0.14
<b>D</b>	11	10	14	18	400	560	1,335	0	0.13	0.20
<b>E</b>	16	13	13	20	479	741	1,668	0.04	0.23	0.47
<b>F</b>	20	20	21	28	1,073	4,254	7,681	1.91	2.30	10.53
<b>G<sup>11</sup></b>	22	23	40	39	1,793	5,837	0	0.18	2.53	8.13
<b>H</b>	24	27	19	37	1,425	3,405	9,210	0	0.79	1.37
<b>I</b>	24	26	36	39	1,730	12,539	15,782	0.01	3.28	11.58
<b>J</b>	27	27	39	43	2,528	22,737	27,296	0.07	8.12	17.92
<b>K</b>	32	34	34	34	2,165	12,585	19,011	0	2.96	5.48
<b>L</b>	37	39	56	50	3,634	38,984	41,500	0.09	18.50	55.85
<b>M</b>	45	49	52	52	3,752	26,128	36,409	0	5.48	14.38
<b>N</b>	56	--	63	67	3,364	52,551	53,699	0.01	26.52	91.07

<sup>11</sup> District G has zero-capacity secondary queues at both stations

Table 2.5 Average problem size and computational results for 14 crew districts

By exhaustively testing the crew assignment model and solution approach for a variety of crew districts with different crew pool sizes, traffic patterns and volumes, and operational configurations (e.g., rotation key, capacities of the secondary queues), we have confirmed that our model and approach performs exceptionally well and that this performance is very robust across districts. In the next section, we further examine the performance of our methods with different problem parameters.

### **2.7.3 Impact of Problem Parameters**

For PSQ districts, the key parameters that may impact the optimal crew assignment decisions are the capacities of secondary queues and costs of crew layover cost and deadheading. In this section, we demonstrate how these parameters can affect the structures of the optimal solutions, using examples from our testing pool. Moreover, we discuss the effects of problem parameters on the performance of our solution methods, and show that our solution methods are robust.

#### ***2.7.3.1 Capacity of secondary queue***

The capacities of secondary queues at both stations (and the rotation key) determine the order in which crews are assigned in PSQ districts, and thus control the workload distributed to each home base. The queue capacities may also impact the effectiveness of some of our enhancements and methods (for example, the latest-connection method) since these methods exploit characteristics of the PSQ rules. Generally, we can achieve more problem reduction with smaller secondary queues because the latest-connection method is able to fix the latest outbound trip for more initial trips when more crews are initially in the primary queue. Table 2.6 demonstrates the

changes in problem reduction, model strengthening, and workload distribution as we change the capacity of secondary queue at one station for a selected instance. When we increase the capacity of the secondary queue at Station A from 80% to 120% of its original value and keep the secondary queue capacity at Station B not changed, we get less reduction in the number of variables and constraints. However, the variation in problem reduction is not significant with a moderate variation in the queue capacity. The improvement in LP bound can be either higher or lower when we increase the capacity of secondary queue because LP bound is not only affected by the distributions of initial crews in the primary and secondary queues, but also the distribution of scheduled trains. As for the workload distribution, more Station B based crews are given higher priority at Station A when we increase the capacity of secondary queue at Station A. Station B based crews are thus assigned more often and have less heldaway time. The ratio of average workload at the two stations decreases from 1.50 to 1.00. The labor unions dynamically change the capacities of secondary queues so that the average workload among the two home bases is balanced in the long run.

Secondary queue capacity at Station A	% problem reduction			% increase in LP bound	Solution		
	x var	RK constr	Push-Pull constr		Workload ratio <sup>12</sup>	Heldaway hours of Station B based crews	CPU time (sec)
<b>Decrease by 20%</b>	42.98	86.74	49.64	17.65	1.50	43.73	18.8
<b>Decrease by 10%</b>	42.27	86.73	47.58	14.02	1.38	38.3	17.82
<b>Original</b>	41.43	86.71	45.41	13.97	1.27	35.01	12.17
<b>Increase by 10%</b>	40.50	86.68	43.14	14.34	1.08	29.44	12.39
<b>Increase by 20%</b>	40.50	86.68	43.36	15.04	1.00	23.91	12.03

<sup>12</sup> Ratio of average per-person workload at home base A and home base B

Table 2.6 Example: impact of secondary queue capacity



Next, we examine the computational performance of our optimization model and the quality of the heuristic solutions, averaged over 10 instances from each district of F to N (districts A – E are not tested because instances from these districts are small and easy to solve), when we change the capacity of the secondary queue at one station. Table 2.7 presents the ranges of the average CPU time and heuristic gaps as we either increase or decrease the capacity of secondary queues at one of the two stations by 20%. The queue capacity at the other station is not changed. The results show that the average CPU time and quality of the heuristic solutions do not vary significantly and do not follow systematic patterns as we change the queue capacities. Therefore, the performance of our solution methods does not depend on the capacity of secondary queues.

District	Avg. CPU time (sec)		Avg. heuristic gap (%)	
	Original	Change queue capacity by 20%	Original	Change queue capacity by 20%
<b>F</b>	2.30	[2.14, 3.03]	1.91	[1.70, 2.01]
<b>G</b>	2.53	[2.50, 3.78]	0.18	[0.12, 0.18]
<b>H</b>	0.79	[0.77, 0.86]	0.00	0.00
<b>I</b>	3.28	[3.29, 4.30]	0.01	[0.00, 0.03]
<b>J</b>	8.12	[8.18, 9.39]	0.07	[0.07, 0.19]
<b>K</b>	2.96	[2.82, 3.21]	0.00	[0.00, 0.29]
<b>L</b>	18.50	[16.22, 22.29]	0.09	[0.07, 0.18]
<b>M</b>	5.48	[5.35, 5.94]	0.00	0.00
<b>N</b>	26.52	[25.53, 32.00]	0.01	[0.01, 0.07]

Table 2.7 Impact of secondary queue capacity on CPU time and heuristic gap

### 2.7.3.2 Deadheading cost

Deadheading is either used to protect against crew shortage or to reduce crew heldaway at the away station. When crews accumulate at one station, we may choose to

either deadhead some crews to the other station or keep them at the station, depending on the relative costs of deadheading and holding crews away from home. Table 2.8 shows the change in heldaway time and number of deadheads in the optimal solution of an instance when we either increase or decrease the deadheading costs by 30%. As expected, we have fewer deadheads and longer heldaway time when deadheading costs increase.

Next, we discuss how deadheading costs may affect the computational performance of our optimization model and the quality of heuristic solutions. Table 2.9 presents the average CPU time and heuristic gap for districts F to K when we increase or decrease deadheading costs by 30%. For most districts, we have slightly longer computational time when deadheading costs are lower. Further, we get a few more instances with positive heuristic gaps when deadheading costs are lower. The slight increases in CPU time and heuristic gaps may result from the fact that the optimal solutions have more deadheads and thus more complex deadheading plans when deadheading is cheaper. But overall, our methods consistently perform well under different cost parameters.

<b>Deadheading costs</b>	<b>Total heldaway time (hour)</b>	<b># of deadheads</b>	<b>CPU time (sec)</b>
<b>Decrease by 30%</b>	108.4	8	3.9
<b>Decrease by 20%</b>	138.7	5	4.17
<b>Decrease by 10%</b>	147.0	4	4.32
<b>Original</b>	147.0	4	3.81
<b>Increase by 10%</b>	190.5	0	3.78
<b>Increase by 20%</b>	190.5	0	3.26

Table 2.8 Example: impact of deadheading cost

District	Avg. CPU time (sec)			Avg. heuristic gap (%)		
	Original	Increase costs	Decrease costs	Original	Increase costs	Decrease costs
<b>F</b>	2.30	2.25	2.46	1.91	1.62	1.68
<b>G</b>	2.53	2.24	2.61	0.18	0.06	0.18
<b>H</b>	0.79	0.76	0.79	0.00	0.00	0.00
<b>I</b>	3.28	2.78	4.87	0.01	0.01	0.24
<b>J</b>	8.12	7.93	10.89	0.07	0.07	0.08
<b>K</b>	2.96	2.95	3.16	0.00	0.00	0.91
<b>L</b>	18.50	21.85	25.37	0.09	0.12	0.07
<b>M</b>	5.48	5.51	6.02	0.00	0.00	0.00
<b>N</b>	26.52	24.86	22.94	0.01	0.01	0.02

Table 2.9 Impact of changing deadheading costs by 30%

The extensive testing results demonstrate that the performance of our solution methods is quite robust when the problem parameters change within moderate ranges. This confirmed the practical use of our model and methods in the real-life operations.

## 2.8 CONCLUSIONS

Effective deployment and utilization of train crews is very important for freight railroads since crew costs constitute a significant proportion of operating expenses. Crew assignment in U.S. freight railways is very different from crew scheduling in airlines or passenger railways due to the many operational differences in these settings. And, unlike crew planning in airlines, the literature on models and methods for freight train crew planning is relatively sparse. Specifically, the literature has not previously modeled or solved crew assignment for double-ended districts with dispatching rules other than simple rotation key rules. This study, motivated by interactions and actual decision support needs at a major U.S. freight railway company, has focused on crew

assignment in double-ended PSQ districts, and provided a mixed-integer programming formulation that accounts for the complex crew dispatching rules in these districts. To solve the problem effectively and quickly, we proposed several enhancements, including model strengthening and problem reduction, that exploit the structure of the constraints and properties of the PSQ rules. In addition, we developed a heuristic algorithm that generates high-quality solutions quickly, and accelerates solution of the exact model. Extensive computational testing and validation of our heuristic method and the exact procedure using our enhanced model demonstrate that our approach generates optimal or near-optimal solution within minutes, making it practical for implementing within a real-time decision support tool for crew dispatchers.

## **Chapter 3: Crew Planning with Uncertainty in Train Arrival and Departure Times**

### **3.1 INTRODUCTION**

In Chapter 2, we discussed crew assignments with the assumption that train schedules are known with certainty in a short planning horizon. This assumption is commonly adopted in many crew scheduling models discussed in the literature. However, trains' departure and arrival times may deviate from the plan due to many factors in real-world operations, especially in the context of freight railroads. For example, the time a train needs to travel from one station to another greatly depends on the level of traffic congestion and whether or not there is a service disruption (such as track maintenance and weather condition). Also, the time required for various work events in a terminal or yard, such as changing crew, fueling, inspection, and picking up or setting out locomotives and cars, is highly variable depending on crew availability and terminal/yard capacity. These factors all impact a train's departure time at the current station and arrival time at the next station.

With significant uncertainty in train arrival and departure times, the optimal deadheading plan from a deterministic model may perform poorly in reality. For instance, a deterministic model may suggest a solution where a crew member from an inbound trip  $i$  is assigned to an outbound trip  $j$  after the minimum required rest. This connection from trip  $i$  to trip  $j$  may become invalid when trip  $i$  arrives later or trip  $j$  is ready to depart earlier than scheduled. In this case, we have to delay trip  $j$  if no other

crew is rested for trip  $j$  and there is no available extra crew. Note that delaying trip  $j$  may have a cascading impact on all later assignments. We could have avoid delaying trip  $j$  if we had incorporated the uncertainty in train arrival/departure time and planed a deadhead from the other station in advance.

This study discusses crew repositioning between two stations, taking into account the uncertainty in train arrival and departure times. In contrast to Chapter 2, this study does not focus on the exact crew assignments and detailed crew dispatching rules. Instead, its goals are to: (i) develop crew planning models and methods that incorporate the uncertainty in trains' arrival and departure time, (ii) demonstrate the value of proactively consider such uncertainty in the planning model, and (iii) provide some insights on crew planning when train schedules are uncertain.

This study is intended as a first-step to incorporate train arrival/departure time uncertainty in crew planning. We consider single-ended crew districts with one home station at which all crew members are based and one away station. At the home station, we can usually use extra crews to operate a train if none of the regular crew members are available. Therefore, our main focus is crew availability at the away station. Using an analogy with inventory problems, crew members arriving at the away station from the home station are (stochastic) supplies. Trains arriving at the away station from an adjacent crew district, which require crew members of this district to take them to the home station, are (stochastic) demands. If crews are rested and waiting to be assigned to a train going back home, there is a cost for lodging and meals as well as an hourly penalty for holding crews away from home. On the other hand, if no crew is rested and

available when a train is ready to depart from the away station to the home station, we have to delay the train, with train delay cost, until a crew member becomes rested. So the main decision of this problem is how many crew members to keep at the away station over time by repositioning crew members between the two stations. In this study, we propose models that balance the costs for crew holding, train delay (due to crew unavailability), and deadheading, given the distributions of arrival times (of trains coming from the home station) and departure times (of trains going from the away station to the home station).

The rest of this chapter is organized as follows. In Section 3.2, we provide a brief review of previous work studying railway/airline crew scheduling with uncertainty in train/flight schedules. Sections 3.3 and 3.4 describe the ingredients and modeling of the crew planning problem that we study. Section 3.5 proposes a model that focuses on inbound deadheading. An exact recursive method and two heuristic algorithms are developed to solve the model. Section 3.6 extends the model and solution approaches to incorporate both inbound and outbound deadheading. We conduct extensive computational experiments in Section 3.7 and conclude in Section 3.8.

## **3.2 LITERATURE REVIEW**

In the airline industry, researchers have proposed two distinct approaches to deal with flight schedule uncertainty that arise in crew scheduling. The first method, sometimes subsumed under “disruption” management, reschedules crews whenever the flight schedules are updated and the original crew schedules become infeasible. The updated flight schedules are then taken as a deterministic input to the crew rescheduling

models. The second method, in contrast, proactively incorporates flight schedule uncertainties in the crew scheduling stage. The resulting crew schedules are expected to be more robust and have better operational performance than those obtained using deterministic flight schedules.

Many researchers have studied the problem of crew rescheduling under disruptions. Most of these studies assume that crew rescheduling is carried out after flights are rescheduled. Consequently, the objective of crew rescheduling is to construct a set of feasible crew pairings that are compatible with the new modified flight schedules and do not severely deviate from the original crew schedules. Clausen et al. (2010) provides a recent survey on airline disruption management, including flight recovery and crew recovery. If the crew rescheduling problem turns out to be infeasible under the new flight schedules, then the flight rescheduling problem is re-solved to provide a different set of flight schedules.

Several recent studies consider the problem of generating robust crew schedules that are more cost-effective for operations with disruptions. A common finding of these studies shows that robust crew pairings tend to have fewer plane changes and longer connection times between flights. Schaefer et al. (2005) propose two methods to approximate the true operating cost of a pairing under disruption. Using the approximate cost they solve the traditional set-covering model of deterministic crew pairing. Their first method evaluates the expected operational cost of a pairing using a Monte Carlo simulator, assuming that crew pairings are independent of each other and planes are always available. The second method penalizes certain pairing patterns that



may lead to poor performance in operations, such as short connection times between plane switches, short rest time between a crew member's consecutive duties, and long flying time of a duty. Yen and Birge (2006) extend the model of Schaefer et al. (2005) to include a recourse function that evaluates the expected extra delay cost due to the interaction between pairings. Their model is a nonlinear two-stage stochastic programming model. To solve the problem, the authors propose a branch-and-bound algorithm that solves the traditional crew pairing problem (Stage 1) at each node of the search tree. Then, the method solves a Stage 2 problem to identify the most expensive flight connection, given the pairing decision at that tree node. Based on the selected flight connection, two branches are created with one branch allowing this connection and the other one forbidding it. Tam et al. (2011) discuss several improvements of the algorithm of Yen and Birge (2006). In particular, they propose a new method to evaluate extra delay due to flight interaction and capture the chain impact of delays.

Ehrgott and Ryan (2002) propose a bi-criteria optimization problem that minimizes the crew pairing cost plus penalties for non-robust flight connections. The problem is solved using the  $\varepsilon$ -constraint method that solves a penalty minimization problem while enforcing that the total pairing cost is within a specified limit  $\varepsilon$ . Shebalov and Klabjan (2006) propose a measure of crew schedule robustness in terms of the flexibility for swapping a crew from one pairing to another pairing. Their model maximizes the total number of possible swaps while limiting the cost of the selected pairings to be within a specified limit. Ionescu and Kliewer (2011) restrict the decisions to crew swaps for flight connections that are likely to propagate delays under disruption.

The swap flexibility is formulated as a recourse function for the traditional crew pairing problem.

In the railway context, we are not aware of any work that explicitly deals with uncertainty in train arrival/departure time for crew scheduling. Gorman and Sarrafzadeh (2000) discuss several heuristic methods to handle uncertainty in train schedules within the framework of a deterministic model. They conclude that simple parameter modifications, such as increasing the minimum rest time, keeping safety stock of crew members, and assuming worst-case train arrival and departure scenarios, generally do not perform well compared to the manual solutions from well-experienced crew planners.

The problem we study is very different from those discussed in the crew scheduling literature for airlines. First, as discussed in Chapter 2, crew planning in U.S. freight railroad is performed for each crew district independently. The problem has simpler network structure than the flight network, and has different cost components and operational rules. Second, unlike in airlines, train crews' consecutive assignments are typically on different trains (in opposite directions). Therefore, we are not concerned with the uncertainty in train dwell time at a station. Further, we can assume that the departure time of an outbound train is independent from the arrival time of an inbound train. In the airline context, it may be necessary to consider the uncertainty in flight ground time. Also, a delay in a flight's arrival time has a cascading effect on the following departure, impacting the feasibility of crew pairings. Finally, our models mainly focus on the planning of deadheads rather than the exact assignment of crew members to trains (or exact crew pairings in the airline context).

### 3.3 PROBLEM DESCRIPTION

In U.S. freight railways, train crews are assigned to individual crew districts and only operate trains between the stations of their assigned district. Depending on the number of stations and crew bases, crew districts are categorized as single-ended, double-ended, or multi-ended districts. In this study, we focus on single-ended districts which have one home station and one away station. Single-ended district is the most common type of crew district and has relatively simple operational rules. Since it only has one crew home base, it does not require complex crew dispatching rules to balance workload among home bases (such as the primary-secondary queue rules discussed in Chapter 2). Instead, it only needs simple rotation key rules such as FIFO or FOFO. At the home station, we assume there are unlimited extra crew members that we can use whenever there is a shortage of regular crews. Therefore, our main focus is to maintain crew availability at the away station.

When train schedules are known with certainty (as discussed in Chapter 2) we can always find a deadheading plan such that at least one crew member is available when a train is ready to depart from the away station to the home station. However, train schedules are subject to uncertainty in the context of freight railroads due to many operational and environmental factors, especially when the planning horizon is beyond one or two days. In this case, if we still want to avoid train delays incurred by crew shortage, we need to keep sufficient crew members at the away station at all times. Such policy requires repositioning many crew members from the home to the away station and incurs unnecessarily high deadheading and crew holding cost. On the other

hand, if we do not proactively deadhead crews to the away station, trains will be delayed until a crew member becomes available because we do not have extra crew members at the away station. Note that train delays have a cascading effect when we staff trains on a first-come-first-serve basis. Deadheading is necessary not only because of the uncertainty in train arrival/departure time, but also due to traffic imbalance. When more trains are going from the home station to the away station than the other way around during a specific period of time, we need to send crews home via deadheading to reduce their idle time at the away station. Hence, the key objective for crew planning is to proactively and carefully reposition crews between the two stations so as to balance the expected crew holding cost, amount of train delay, and crew deadheading expenses.

Crews normally need to take a rest between two consecutive assignments. There are different requirements for the length of rest depending on the length of the assignments before and after the rest. In this study, we assume that crews always need to take a full rest (typically 10 hours) between assignments. Resting at the home station has no cost to the railroad. At the away station, cost during the full rest period is considered as sunk cost and is thus ignored. After that, the railroad incurs *crew holding cost*, representing the cost for lodging and per hour payments to the crew members for the time being held away from home beyond a certain threshold. We use a linear function to approximate the crew holding cost such that it is proportional to the holding time after crew becoming fully rested. To penalize train delay due to crew shortage, we impose artificial *train delay cost* that is proportional to the length of delay. We assume the delay cost per time unit is the same for all trains, regardless of trains' priorities and

scheduled departure times. For crew repositioning, we only consider public transit modes, such as buses, shuttles, and air flights, whose schedules are independent of those of the freight trains. **Deadheading cost** is incurred as a per-person cost. We determine all deadheads at the beginning of planning horizon and assume that deadheads cannot be cancelled once they are scheduled.

Trains arriving at the away station from the home station are called *inbound trains*, and those departing from the away station to the home station are called *outbound trains*. The actual arrival time of inbound trains and departure times of outbound trains are unknown at the time of planning. However, we may obtain the distribution of a train's arrival/departure time from the analysis of historical data. Further, we assume the arrival and departure times of different trains are independent.

Finally, we only consider one occupation type, say engineer, for simplicity. In fact, crew members of different occupation types usually have similar work schedules in practice, especially in single-ended districts.

Below we summarize the problem setting and assumptions:

- Single-ended crew district
- Full rest at the away station
- Unlimited extra crews at the home station
- Linear train delay cost and crew holding cost at the away station; The cost is the same for all trains/crews
- Per-person deadheading cost
- Known independent distributions of train arrival/departure times

- Deadheading trips' schedules are independent of the schedules of freight trains
- Deadheads cannot be cancelled once planned at the beginning of horizon

In the next section, we set up the problem as a multi-period inventory control problem that maintains crew availability at the away station. We propose models and solution methods that minimize the total expected cost over a given planning horizon, including expected crew holding cost, expected train delay cost, and deadheading cost. Note that the models do not provide exact crew-to-train assignments and thus do not explicitly impose the rotation key constraints. However, given a deadheading plan proposed by the models and a realization of train arrivals and departures, we can easily generate crew-to-train assignments that satisfy the rotation key constraints.

### 3.4 PROBLEM MODELING

For a given planning horizon, we have a set of trains that are scheduled to arrive at the away station from the home station (inbound) or depart from the away station to the home station (outbound). We divide the planning horizon into  $T$  periods. The problem of maintaining crew availability at the away station mimics a finite-horizon multi-period inventory control problem. Crews arriving at the away station from the home station are supplies. Each inbound train brings one crew member, or one unit of *regular supply*. Inbound deadheading trips may bring multiple crews that are called *additional supplies*. Similarly, each train departing from the away station to the home station is one unit of *regular demand*, whereas outbound deadheading trips that send spare crews back home are called *additional demands*. Regular supply becomes available when a crew becomes rested. Additional supplies has a lead time that equals to the transit time of

deadheading trip from the home station to the away station plus the time needed for the crew to get fully rested. We use  $l$  to denote the first period when an additional supply will be rested and available to be assigned.

For each period, we create one inbound and one outbound candidate deadheading trip. Let  $i_t, t = l, \dots, T$  be the candidate inbound deadheading trip whose supply will *get rested in period  $t$* , and  $j_t, t = 1, \dots, T$  be the candidate outbound deadheading trip that departs from the away station in period  $t$ . Assume deadheading trips are uncapacitated. Each deadhead incurs a per-person cost  $C$ .

Holding cost is  $H$  per person per period if a crew member is rested but not assigned to an outbound trip. Cost for delaying a train (backorder) due to crew unavailability is  $B$  per train per period. Note that the demands are homogeneous in terms of cost of delay. If more than one train is waiting for an available crew member, we can assign crews to these trains in any order. For example, we may either assign the first rested crew to the train that arrived earliest or the one with the highest priority.

In this study, we focus on static-decision making models which provide a deadheading plan for the entire planning horizon at the beginning of period one. The decision variables are  $x_t$  and  $y_t$ , which are the number of crews deadheaded on inbound trip  $i_t$  and outbound trip  $j_t$ , respectively. We also define  $X_t$  as the cumulative number of inbound deadheads rested up to period  $t$  ( $X_t = \sum_{i \leq t} x_i$ ), and  $Y_t$  as the number of cumulative outbound deadheads up to period  $t$  ( $Y_t = \sum_{i \leq t} y_i$ ). Let  $S_t$  denote the cumulative regular supply up to period  $t$ , i.e., the number of crew members who arrive on

inbound trains and become rested on or before period  $t$ . Let  $D_t$  be the cumulative regular demand in periods 1 to  $t$ .  $S_t$  and  $D_t$  are random variables that depend on the distributions of trains' schedules.

In Sections 3.5 and 3.6 we discuss two models for deadhead planning. The first model only considers inbound deadheading to avoid or reduce crew shortage at the away station, while the second model simultaneously considers both inbound and outbound deadheading options. We provide below a summary of the notation needed to model the problem.

**Notation:**

$T$ : number of periods in the planning horizon

$l$ : index of the first period when additional supply is available

$\tau$ : length of one period

$I, J$ : sets of inbound and outbound trains, respectively

$p_{it}, p_{jt}$ : probabilities of crews from inbound train  $i$  getting rested in period  $t$  and outbound train  $j$  departing in period  $t$ , for  $i \in I, j \in J, t = 1, \dots, T$

$i_t$ : candidate inbound deadheading trip whose supply becomes rested in period  $t, t = l, \dots, T$

$j_t$ : candidate outbound deadheading trip departing in period  $t, t = 1, \dots, T$

$x_t$ : number of crew members deadheaded on trip  $i_t$ ;  $\mathbf{x} = (x_1, \dots, x_T)$ ;  $x_1 = \dots = x_{l-1} = 0$

$y_t$ : number of crew members deadheaded on trip  $j_t$ ;  $\mathbf{y} = (y_1, \dots, y_T)$

$X_t$ : cumulative additional supply rested up to period  $t$ ;  $X_t = \sum_{t'=1}^t x_{t'}$ ;  $\mathbf{X} = (X_1, \dots, X_T)$

$Y_t$ : cumulative additional demand up to period  $t$ ;  $Y_t = \sum_{t'=1}^t y_{t'}$ ;  $\mathbf{Y} = (Y_1, \dots, Y_T)$

$S_t$ : cumulative regular supply rested up to period  $t, t = 1, \dots, T$

$D_t$ : cumulative regular demand up to period  $t, t = 1, \dots, T$

$\hat{D}_t$ : net cumulative regular demand up to period  $t, t = 1, \dots, T$ ;  $\hat{D}_t = D_t - S_t$

$\Phi_t$ : cumulative distribution function (CDF) of net demand  $\hat{D}_t$

$\Omega$ : set of scenarios of inbound train arrivals and outbound train departures; indexed by  $\omega$

$H$ : holding cost per crew member per period

$B$ : delay cost per train per period



$C$ : per-person deadheading cost

$\rho$ : critical ratio defined as  $\rho = B/(H + B)$

### 3.5 MODEL 1: INBOUND DEADHEADING

In this model, we want to find the optimal plan of inbound deadheading, determined at time 0, that minimizes the total deadheading, expected crew holding and expected train delay costs. This problem mimics a multi-period inventory problem where we want to find an ordering plan that minimizes ordering, inventory holding, and backorder costs. In this section, we first develop the formulation of the problem, and then propose two solution approaches. The first approach is an exact recursive method while the second one is a heuristic based on newsvendor ordering policy.

At time zero, we decide the number of crew members to be assigned on each candidate deadheading trip. For a given realization of train arrivals and departures, we can easily compute crew holding and train delay costs. The expected cost is the average cost under all possible scenarios of train arrivals and departures. Suppose  $\Omega$  is the set of all possible scenarios. For a given scenario  $\omega \in \Omega$ , the cumulative regular supply and regular demand are  $S_t^\omega$  and  $D_t^\omega$ , respectively. With a given deadheading plan, i.e., a set of values of  $x_t$  and  $X_t$  for all  $t$ , crew holding cost is  $H(S_t^\omega + X_t - D_t^\omega)^+$  and train delay cost is  $B(D_t^\omega - S_t^\omega - X_t)^+$  for each period  $t$ . The total cost under scenario  $\omega$  is thus  $\sum_{t=1}^T [H(S_t^\omega + X_t - D_t^\omega)^+ + B(D_t^\omega - S_t^\omega - X_t)^+]$ . The expected cost under all possible scenarios is as follows:

$$\begin{aligned}
& E_{\omega} \sum_{t=1}^T \left[ H \left( S_t^{\omega} + X_t - D_t^{\omega} \right)^+ + B \left( D_t^{\omega} - S_t^{\omega} - X_t \right)^+ \right] \\
& = \sum_{t=1}^T E_{\omega} \left[ H \left( S_t^{\omega} + X_t - D_t^{\omega} \right)^+ + B \left( D_t^{\omega} - S_t^{\omega} - X_t \right)^+ \right].
\end{aligned}$$

Cost of the given deadheading plan is  $\sum_{t=l}^T Cx_t$ .

For notational brevity, we omit the script  $\omega$ . Also, let  $\hat{D}_t = D_t - S_t$  be the *net cumulative regular demand* in period  $t$ . We formulate the inbound only crew planning problem **[IBCP]** in (3.1).

$$\begin{aligned}
\textbf{[IBCP]} \quad & \text{minimize} \quad \sum_{t=l}^T E \left[ H(X_t - \hat{D}_t)^+ + B(\hat{D}_t - X_t)^+ \right] + Cx_t \\
& \text{s.t.} \quad x_t \geq 0, \forall t = l, \dots, T
\end{aligned} \tag{3.1}$$

Note that the objective function only minimizes the costs in periods  $l$  to  $T$  because the costs in periods 1 to  $(l - 1)$  are sunk cost that do not depend on the deadheading decisions.

### 3.5.1 Recursive method for inbound deadheading

Let  $F_t(X_t)$  be the minimum cumulative cost up to period  $t$  (inclusive) when  $X_t$  additional supplies are ordered and have become rested on or before period  $t$ . The deadheading cost in period  $t$  is  $Cx_t$  (we collect deadheading cost in the period when the crew become rested). The expected crew holding and train delay costs in period  $t$  are  $H * E(X_t - \hat{D}_t)^+$  and  $B * E(\hat{D}_t - X_t)^+$ , respectively. Equation (3.2) states the recursive relationship between period  $t-1$  and period  $t$ . The initial condition is  $F_{l-1}(X_{l-1} = 0) = 0$ , and the optimal value function is  $\min_{X_T} F_T(X_T)$ .

**Recursive function:**

$$F_t(X_t) = \min_{x_t} F_{t-1}(X_{t-1}) + Cx_t + E\left[H(X_t - \hat{D}_t)^+ + B(\hat{D}_t - X_t)^+\right], \forall t = 1, \dots, T \quad (3.2)$$

The feasible region of decision variable  $X_t$  is bounded above by the maximum number of crew we need to possibly deadhead on trip  $i_t$ . Since deadheading cost is uniform across periods, we do not need to consider deadheading in period  $t$  for potential crew shortage in later periods. Therefore,  $0 \leq X_t \leq \left[\hat{D}_t^{\max}\right]^+$ , where  $\hat{D}_t^{\max}$  is the maximum possible net regular demand in period  $t$ . Also, we need  $X_t \geq X_{t-1}$ . Therefore, the upper bound of  $X_t$  is  $X_t^{\max} = \max_{t' \leq t} \left[\hat{D}_{t'}^{\max}\right]^+$ . For a given value of  $X_t$ , recursive function (3.2) needs to enumerate all possible values of  $x_t$  in the range of  $[0, X_t]$  in order to find the optimal  $x_t$  value. Therefore, the computational complexity of this algorithm is  $O\left(T \cdot (X_t^{\max})^2\right)$ .

**3.5.2 Newsvendor-based heuristic for inbound deadheading**

In this section, we propose a heuristic that embeds the idea of newsvendor ordering policy. Our problem of inbound deadhead planning at the away station mimics a multi-period inventory problem with net demands  $\hat{D}_t$ . The distribution (CDF) of  $\hat{D}_t$  is  $\Phi_t(a) = P(\hat{D}_t \leq a), a \in (-\infty, +\infty)$ . Note that, in contrast to the demand in traditionally inventory problem,  $\hat{D}_t$  could be positive, negative, or zero, and the sequence of net demands is not necessarily stochastically increasing. In Section 3.5.2.1, we first develop the optimal solution of a multi-period inventory problem with a few

assumptions, including continuous supply and demand, and stochastic dominance in  $\hat{D}_t$ .

Then we propose a heuristic procedure in Section 3.5.2.2 to solve our original problem with discrete supply and demand. Section 3.5.2.3 modifies the base heuristic of Section 3.5.2.2 to better handle the fact that  $\hat{D}_t$  is not necessarily stochastically increasing.

### 3.5.2.1 Optimal solution for a continuous multi-period inventory problem

In this section, we derive the structure of the optimal ordering policy for problem (3.1) with the following assumptions. We call this altered problem *continuous-IBCP*.

- (i)  $x_t$  is a continuous decision variable in the support of  $[0, +\infty]$ ;
- (ii)  $\hat{D}_t$  is a continuous random variable in the support of  $[-\infty, +\infty]$ ;
- (iii)  $\Phi_t$  has continuous first-order derivatives;
- (iv) The sequence of  $\hat{D}_t$  is stochastically increasing, i.e.,  $\Phi_t(a) \geq \Phi_{t+1}(a)$  for any  $a \in (-\infty, +\infty)$ , or equivalently,  $\Phi_{t+1}^{-1}(\sigma) \geq \Phi_t^{-1}(\sigma)$  for any  $\sigma \in [0, 1]$ .

Let  $\mu_t$  be the Lagrange multiplier corresponding to constraint  $x_t \geq 0$ ,  $\forall t = l, \dots, T$ . The Lagrange function of *continuous-IBCP* is as follows:

**Lagrange function:**

$$L(\mathbf{x}, \boldsymbol{\mu}) = \sum_{t=l}^T \mathbb{E} \left[ H(X_t - \hat{D}_t)^+ + B(\hat{D}_t - X_t)^+ \right] + (C - \mu_t)x_t. \quad (3.3)$$

If  $\mathbf{x}^*$  is an optimal solution to *continuous-IBCP* and  $\mathbf{X}^*$  is the corresponding optimal cumulative deadhead count, then there exists  $\boldsymbol{\mu}^*$  that satisfies the following KKT conditions:

**KKT conditions:**

- (1) Primal feasibility:  $\mathbf{x}^* \geq 0$ ;
- (2) Dual feasibility:  $\boldsymbol{\mu}^* \geq 0$ ;
- (3) Stationarity:

$$\frac{d}{dx_t} L(\mathbf{x}, \boldsymbol{\mu}) \Big|_{\mathbf{x}=\mathbf{x}^*} = \sum_{i=t}^T \left[ (H+B)\Phi_i(X_i^*) - B \right] + C - \mu_t^* = 0, \forall t = l, \dots, T;$$

- (4) Complementary slackness:  $\mu_t^* x_t^* = 0, \forall t = l, \dots, T$ .

The stationarity conditions can be re-written as follows:

$$\begin{aligned} \mu_t^* &= (H+B)\Phi_t(X_t^*) - B + \mu_{t+1}^*, \forall t = l, \dots, T-1, \\ \mu_T^* &= (H+B)\Phi_T(X_T^*) - B + C. \end{aligned}$$

**Lemma 3.1:**

If there exists an integer  $k \in \{l+1, \dots, T\}$  such that  $\mu_{k-1}^* = 0$  and  $\mu_k^* > 0$ , then  $\mu_t^* > 0$  for all  $t \geq k$ .

This Lemma is briefly proved in Chen and Graves (2013). For completeness, we show the proof below.

**Proof:** According to the stationarity condition,  $\mu_{k-1}^* = (H+B)\Phi_{k-1}(X_{k-1}^*) - B + \mu_k^*$ .

Given that  $\mu_{k-1}^* = 0$  and  $\mu_k^* > 0$ , we have  $(H+B)\Phi_{k-1}(X_{k-1}^*) - B < 0$ . Next, we show

that if  $(H+B)\Phi_{t-1}(X_{t-1}^*) - B < 0$  and  $\mu_t^* > 0$ , then  $(H+B)\Phi_t(X_t^*) - B < 0$ , for  $k \leq t \leq$

$T-1$ , and  $\mu_{t+1}^* > 0$ . From complementary slackness,  $\mu_t^* > 0$  implies  $x_t^* = 0$ , and thus

$X_t^* = X_{t-1}^* + x_t^* = X_{t-1}^*$ . With stationarity,  $\mu_t^* = (H+B)\Phi_t(X_t^*) - B + \mu_{t+1}^*$ , we have

$\mu_t^* - \mu_{t+1}^* = (H+B)\Phi_t(X_t^*) - B = (H+B)\Phi_t(X_{t-1}^*) - B \leq (H+B)\Phi_{t-1}(X_{t-1}^*) - B$ . The last

inequality is valid because we have assumed the sequence of  $\hat{D}_t$  is stochastically

increasing. Given that  $(H+B)\Phi_{t-1}(X_{t-1}^*)-B < 0$ , we have  $\mu_{t+1}^* > \mu_t^* > 0$  and  $(H+B)\Phi_t(X_t^*)-B < 0$ . Therefore, by the principle of induction we have  $\mu_T^* > \dots > \mu_k^* > 0$ . ♦

**Corollary 3.1:**

There exist integers  $n \in \{l-1, \dots, T\}$  and  $m \in \{l, \dots, T+1\}$ ,  $n < m$ , such that the Lagrange multiplier  $\mu_t^* > 0$  for  $t \leq n$  or  $t \geq m$ , and 0 otherwise.

**Proof:** Dual feasibility requires  $\mu^* \geq 0$ . Therefore, the values of  $\mu_t^*$  must have one of the following forms:

- (i)  $\mu^* > 0$ ;
- (ii)  $\mu^* = 0$ ;
- (iii) There exist  $\mu_i^* = 0$  and  $\mu_j^* > 0$ , but not  $m \in \{l+1, \dots, T\}$  such that  $\mu_{m-1}^* = 0$  and  $\mu_m^* > 0$ ;
- (iv) There exists  $m \in \{l+1, \dots, T\}$  such that  $\mu_{m-1}^* = 0$  and  $\mu_m^* > 0$ .

Case (i) is a special form of  $\mu^*$  with either  $n = T$  or  $m = l$ .

Case (ii) is a special form of  $\mu^*$  with  $n = l-1$  and  $m = T+1$ .

In the third case, let  $n+1$  ( $n \geq l$ ) be the minimum index of zero-valued Lagrange multipliers, i.e.,  $\mu_t^* > 0, \forall t \leq n$  and  $\mu_{n+1}^* = 0$ . Since there is no  $m \in \{l+1, \dots, T\}$  such that  $\mu_{m-1}^* = 0$  and  $\mu_m^* > 0$ , we must have  $\mu_{t+1}^* = 0$  if  $\mu_t^* = 0$ . Therefore,  $\mu_t^* = 0, \forall t \geq n+1$ . In this case, we can set  $m = T+1$ .

In the last case,  $\mu_{m-1}^* = 0$  and  $\mu_m^* > 0$  imply that  $\mu_t^* > 0$  for all  $t \geq m$  (Lemma

3.1). The values of  $\mu_t^*$ ,  $t \leq m-2$ , can be one of the following scenarios: (1) all values are positive, i.e.,  $\mu_t^* > 0, \forall t \leq m-2$  (in this scenario  $n = l-1$ ); (2) there is at least one multiplier equals 0 and  $n+1$  is the minimum index where  $\mu_{n+1}^* = 0$ , i.e.,  $\mu_t^* > 0, \forall t \leq n$ . Then we must have  $\mu_{n+1}^* = \dots = \mu_{m-2}^* = 0$ ; otherwise  $\mu_{m-1}^* > 0$  (Lemma 3.1), which contradicts with the given condition  $\mu_{m-1}^* = 0$ .

To sum up,  $\mu^*$  has the general structure of  $\mu_t^* > 0$  for all  $t \leq n$  and  $t \geq m$ , and 0 otherwise, where  $l-1 \leq n \leq T$ ,  $l \leq m \leq T+1$ , and  $n < m$ . ♦

**Theorem 3.1:**

The optimal solution to *continuous-IBCP* has one of the following forms:

- (1)  $\mathbf{x}^* = 0$
- (2) Let  $k$  be the last ordering period (i.e.,  $x_k^* > 0$ , and  $x_t^* = 0, \forall t > k$ ). The total ordering quantity  $X_k^*$  satisfies equation (3.4), and  $X_k^* > [\Phi_{k-1}^{-1}(\rho)]^+$  if  $k > l$  or  $X_k^* > 0$  if  $k = l$ .

$$C - B(T - k + 1) + (H + B)[\Phi_k(X_k^*) + \dots + \Phi_T(X_k^*)] = 0. \quad (3.4)$$

The optimal deadheading plan is:

$$\begin{aligned} x_t^* &= 0, \forall t = 1, \dots, l-1 \\ x_t^* &= \left[ \Phi_t^{-1}(\rho) - \sum_{i=1}^{t-1} x_i^* \right]^+, t = l, \dots, k-1 \\ x_k^* &= \left[ X_k^* - \sum_{i=1}^{k-1} x_i^* \right]^+ \\ x_{k+1}^* &= \dots = x_T^* = 0. \end{aligned}$$

**Proof:** Collorary 3.1 indicates that  $\mu^*$  has five possible structures depending on the values of  $n$  and  $m$ . We can derive  $\mathbf{x}^*$  from the KKT conditions for each case of  $\mu^*$ .

Case 1: There exists  $n \in \{l, \dots, T-1\}$  and  $m \in \{l+1, \dots, T\}$ ,  $n \leq m-2$  such that

$$\mu_t^* > 0, \forall t \leq n \text{ and } t \geq m, \text{ and } \mu_t^* = 0, \forall t \in \{n+1, \dots, m-1\}.$$

From complementary slackness,  $x_t^* = 0, \forall t \leq n$  or  $t \geq m$ , indicating  $X_n^* = 0$  and

$$X_{m-1}^* = \dots = X_T^*.$$

Plug  $\mu_t^* = 0, \forall n+1 \leq t \leq m-1$  and  $\mu_m^* > 0$  in the stationarity

conditions, we have  $(H+B)\Phi_t(X_t^*) - B = 0, \forall t = n+1, \dots, m-2$ , and

$$(H+B)\Phi_{m-1}(X_{m-1}^*) - B = -\mu_m^* < 0. \text{ Hence, } X_t^* = \Phi_t^{-1}(\rho), \forall t = n+1, \dots, m-2,$$

and  $X_{m-1}^* < \Phi_{m-1}^{-1}(\rho)$ . Because  $\mu_{m-1}^* = 0$  and  $X_{m-1}^* = \dots = X_T^*$ ,  $X_{m-1}^*$  satisfies

equation (3.5):

$$C - B(T - m + 2) + (H + B)[\Phi_{m-1}(X_{m-1}^*) + \dots + \Phi_T(X_{m-1}^*)] = 0. \quad (3.5)$$

To be primal feasible, we must have  $X_{m-1}^* \geq \Phi_{m-2}^{-1}(\rho) \geq \dots \geq \Phi_{n+1}^{-1}(\rho) \geq 0$ .

Because we assume that  $\hat{D}_t$  is stochastically increasing, this condition reduces

to  $\Phi_{n+1}^{-1}(\rho) \geq 0$  and  $X_{m-1}^* \geq \Phi_{m-2}^{-1}(\rho)$ , if  $n < m-2$ ; or  $X_{m-1}^* \geq 0$  if  $n = m-2$ . Next

we check the stationarity conditions for  $t \leq n$ . Plugging in  $X_t^* = \dots = X_n^* = 0$

and  $\mu_{n+1}^* = 0$  yields  $\mu_t^* = \sum_{i=t}^n [(H+B)\Phi_i(0) - B], \forall t \leq n$ . Therefore, to ensure

$$\mu_t^* > 0, \forall t \leq n, \text{ we must have } \mu_n^* = (H+B)\Phi_n(0) - B > 0, \text{ or equivalently,}$$

$$\Phi_n(0) > \rho. \text{ Finally, we check the validity of the stationarity condition for}$$

$t \geq m$ . Plug  $X_T^* = \dots = X_m^* = X_{m-1}^*$ , to the stationarity condition, we have

$$\mu_t^* = C - B(T - t + 1) + (H + B)[\Phi_t(X_{m-1}^*) + \dots + \Phi_T(X_{m-1}^*)], \forall t \geq m. \text{ According to}$$



equation (3.5) and  $X_{m-1}^* < \Phi_{m-1}^{-1}(\rho)$ , the following inequality is true for  $t \geq m$ :

$$C - B(T - t + 1) + (H + B) \left[ \Phi_t(X_{m-1}^*) + \cdots + \Phi_T(X_{m-1}^*) \right] > 0. \quad \text{This agrees with the}$$

fact that  $\mu_t^*, \forall t \geq m$ . Thus, the stationarity condition is valid for  $t \geq m$ .

Case 2: There exists  $n \in \{l, \dots, T-1\}$  such that  $\mu_t^* > 0, t \leq n$  and  $\mu_t^* = 0, t > n$ .

The solution  $(\mathbf{x}^*, \boldsymbol{\mu}^*)$  can be derived in the same way as we did for Case 1 by

setting  $m - 1 = T$ . Note that  $\mu_T^* = C - B + (H + B)\Phi_T(X_T^*) = 0$  indicates that

$$X_T^* = \Phi_T^{-1} \left( \frac{B - C}{H + B} \right).$$

Case 3: There exists  $m \in \{l+1, \dots, T\}$  such that  $\mu_t^* = 0, t < m$  and  $\mu_t^* > 0, t \geq m$ .

The solution  $(\mathbf{x}^*, \boldsymbol{\mu}^*)$  can be derived in the same way as we did for Case 1 by

setting  $n + 1 = l$ .

Case 4:  $\boldsymbol{\mu}^* = 0$ .

By setting  $n + 1 = l$  and  $m - 1 = T$ , we can derive the optimal solution as

$$X_t^* = \Phi_t^{-1}(\rho), \forall t = l, \dots, T-1, \text{ and } X_T^* = \Phi_T^{-1} \left( \frac{B - C}{H + B} \right).$$

Case 5:  $\boldsymbol{\mu}^* > 0$ .

Plug  $\boldsymbol{\mu}^* > 0$  to the complementary slackness conditions, we have  $\mathbf{x}^* = 0$ .

To sum up,  $\mathbf{X}^*$  either equals zero or has the structure of  $X_l^* = \cdots = X_n^* = 0$ ,

$X_t^* = \Phi_t^{-1}(\rho) \geq 0, \forall t = n+1, \dots, m-2$ , and  $X_{m-1}^* = \cdots = X_T^*$ , where  $X_{m-1}^*$  is the solution

to equation (3.5) and  $X_{m-2}^* \leq X_{m-1}^* < \Phi_{m-1}^{-1}(\rho)$ . As discussed in Case 5, stationarity

implies  $\Phi_n(0) > \rho$ , i.e.,  $\Phi_n^{-1}(\rho) < 0$ . Therefore, we can re-write  $\mathbf{X}^*$  in the equivalent form of  $X_t^* = [\Phi_t^{-1}(\rho)]^+, \forall t \leq m-2$ , and  $X_{m-1}^* = \dots = X_T^*$ . Note that if  $X_{m-1}^*$  reaches the lower bound  $X_{m-2}^*$ , the ordering quantity in period  $m-1$  is 0 ( $x_{m-1}^* = 0$ ). Now let  $k \leq m-1$  be the last period with positive ordering quantity, then  $X_k^*$  is a solution to equation (3.4) because  $X_t^* = \Phi_t^{-1}(\rho), \forall k \leq t \leq m-2$ . The optimal deadheading plan is as follows:

$$x_t^* = 0, \forall t < l \text{ or } t > k,$$

$$x_t^* = X_t^* - X_{t-1}^* = [\Phi_t^{-1}(\rho)]^+ - \sum_{i=1}^{t-1} x_i^* = [\Phi_t^{-1}(\rho) - \sum_{i=1}^{t-1} x_i^*]^+, \forall t = l, \dots, k-1, \text{ and}$$

$$x_k^* = [X_k^* - \sum_{i=1}^{k-1} x_i^*]^+. \quad \blacklozenge$$

Note that in the special case of  $\hat{D}_t \geq 0, \forall t = 1, \dots, T$ , *continuous-IBCP* reduces to a traditional multi-period inventory problem with per-unit ordering cost  $C$ . Then the optimal solution is either to order nothing (i.e.,  $\mathbf{x}^* = 0$ ) or order up to period  $k$  where  $X_t^* = \Phi_t^{-1}(\rho), \forall l \leq t \leq k-1$ , and  $X_{k-1}^* < X_k^* \leq \Phi_k^{-1}(\rho)$ .

### 3.5.2.2 Base heuristic for problem with discrete supply and demand

Now we relax the assumptions of continuous variables and continuous distribution function. When supply and demand are discrete, KKT conditions no longer apply. Therefore, the solution procedure discussed in this section is a heuristic method rather than an exact method. The solution procedure proposed below first uses bi-section to find the last ordering period  $k$  and total deadhead count  $X_k^*$ , and then

calculates  $X_t^*, \forall t = l, \dots, k-1$  based on  $\Phi_t^{-1}(\rho)$ . In general,  $\hat{D}_t$  is not stochastically increasing. Therefore, the resulting sequence of  $X_t^*$  may not be monotonically increasing. In this case, we need to adjust  $X_t^*$  so that it is at least as large as  $X_{t-1}^*$ , i.e.,

$$X_t^* := \max\{X_{t-1}^*, X_t^*\}, \forall l \leq t \leq k.$$

### Heuristic 1: base\_NV:

*Define:*  $\lceil \Phi_t^{-1}(\rho) \rceil = a$ , s.t.,  $P(\hat{D}_t \leq a) \geq \rho$  and  $P(\hat{D}_t \leq a-1) < \rho$ ;

$$f_t(a) = \Phi_t(a) + \dots + \Phi_T(a);$$

$$\gamma_k = [B(T-k+1) - C]/(H+B);$$

$$UB_k = \lceil \Phi_k^{-1}(\rho) \rceil;$$

$$LB_k = \begin{cases} \lceil \Phi_{k-1}^{-1}(\rho) \rceil, & \text{if } k > l \\ 0, & \text{if } k = l \end{cases}.$$

Let  $Q^*$  be the smallest integer such that  $f_k(Q^*) \geq \gamma_k$  and  $LB_k < Q^* \leq UB_k$ .

Initialize  $k := T$ .

*Step 1:* Compute  $\gamma_k$ ,  $UB_k$ , and  $LB_k$  using the definition above.

*Step 2 (feasibility):* If  $f_k(LB_k) \geq \gamma_k$ , let  $k := k - 1$ . If  $k = l - 1$ ,  $\mathbf{x}^* = 0$ , stop. Otherwise, repeat Step 1. If  $f_k(LB_k) < \gamma_k$ , go to Step 3.

*Step 3 (stopping criteria):* If  $f_k(LB_k + 1) \geq \gamma_k$ , let  $Q^* = LB_k + 1$ , go to Step 5; Else if  $f_k(UB_k - 1) < \gamma_k$ , let  $Q^* = UB_k$ , go to Step 5. Otherwise, go to Step 4.

*Step 4 (bi-section):* Let  $a = \lceil (LB_k + UB_k)/2 \rceil$ . If  $f_k(a) \geq \gamma_k$ , update  $UB_k := a$ ; else, update  $LB_k := a$ . Go to Step 3.

*Step 5 (compute  $\mathbf{X}^*$ ):*  $X_t^* = 0, \forall t < l$ ;  $X_t^* = \lceil \lceil \Phi_t^{-1}(\rho) \rceil \rceil^+, \forall l \leq t \leq k-1$ ;  $X_k^* = \lceil Q^* \rceil^+$ ; and  $X_t^* = X_k^*, \forall t > k$ .

*Step 6 (adjust  $\mathbf{X}^*$ ):* Let  $X_t^* := \max\{X_{t-1}^*, X_t^*\}, \forall l \leq t \leq k$ . Deadhead orders in each period is  $x_t^* = X_t^* - X_{t-1}^*, \forall l \leq t \leq k$ .

The computational complexity of this solution procedure is  $O(T + \log_2 D_T^{\max})$ .

### 3.5.2.3 Modified heuristic for problem with discrete supply and demand

The base\_NV heuristic tends to order more deadheads and incur higher crew holding cost compared to the optimal solution, especially when the sequence of  $\mathbf{X}^*$  from Step 5 is not monotonically increasing. In particular, we may over order deadheads when a period  $t$  has high net demand but its following periods have low (or even negative) net demands. In general, the initial solution  $\mathbf{X}^*$  obtained from Step 5 of the base\_NV heuristic shows one or more peaks and valleys. For each peak, we have a potential risk of over-ordering. Therefore, we propose a procedure to reduce peaks in  $\mathbf{X}^*$  and obtain a better solution. Suppose  $[X_a^*, X_b^*]$  is a peak, i.e.,  $X_a^* = X_{a+1}^* = \dots = X_b^*$ ,  $X_a^* > X_t^*, \forall t < a$ , and  $X_b^* > X_{b+1}^*$ . Let  $t_1$  be a later period with  $X_{t_1}^* \geq X_b^*$  or  $t_1 = T+1$  if  $X_t^* < X_b^*, \forall t > b$ . We use  $\tilde{\mathbf{X}}^*$  to denote the sequence of cumulative ordering quantity using Step 6 of the base\_NV heuristic. If we lower the peak  $[X_a^*, X_b^*]$  by one, then  $\tilde{X}_t^* := \tilde{X}_t^* - 1, \forall a \leq t < t_1$ ;  $\tilde{X}_t^*$  remains unchanged for the other periods. Therefore, reducing the peak value by one only affects the expected crew holding and train delay costs for periods from  $a$  to  $t_1 - 1$ . We expect to have lower crew holding cost but higher train delay cost in those periods. If  $X_{t_1}^* \geq X_b^*$ , then the deadhead in period  $a$  is deferred to period  $t_1$ ; otherwise, reducing the peak value by one essentially cancels one deadhead in period  $a$  and thus saves one person's deadheading cost. If the total expected saving in crew holding cost (and deadheading cost) is higher than the expected increase in train delay cost, we can obtain a better solution by setting

$\tilde{X}_t^* := \tilde{X}_t^* - 1, \forall a \leq t < t_1$ . If the expected cost saving is lower than the increase in train delay cost, we move on to the next peak and apply the same procedure to determine whether it is beneficial to lowering the peak. We can continue this procedure until there are no more peaks that should be lowered. We present the details of this modified heuristic below.

### Heuristic 2: reduce\_peak:

*Step 1* (initialize  $\mathbf{X}^*$ ): Compute the initial sequence of  $\mathbf{X}^*$  using Step 1 to Step 5 of the base\_NV heuristic, i.e.,  $X_t^* = \left\lceil \left\lceil \Phi_t^{-1}(\rho) \right\rceil \right\rceil^+, \forall l \leq t \leq k-1$ ;  $X_t^* = \left\lceil Q^* \right\rceil^+, \forall t \geq k$ .

Let  $t_0 = l$ .

*Step 2* (find peak): Search for a peak starting from  $t_0$ . If no more peaks are found, go to Step 5. Otherwise, let  $[X_a^*, X_b^*]$  be the first peak since  $t_0$ , i.e.,  $b \geq a \geq t_0$ ,  $X_a^* = X_{a+1}^* = \dots = X_b^*$ ,  $X_a^* > X_t^*, \forall t < a$ , and  $X_b^* > X_{b+1}^*$ . Go to Step 3.

*Step 3* (calculate marginal costs): Let  $t_1$  be the first period after period  $b$  with  $X_{t_1}^* \geq X_b^*$  or  $t_1 = T+1$  if  $X_t^* < X_b^*, \forall t > b$ . Let  $\Delta^-$  be the marginal benefit and  $\Delta^+$  be the marginal cost from period  $a$  to  $t_1 - 1$  when we reduce one deadhead in period  $a$ :

$$\Delta^- = H \sum_{t'=a}^{t_1-1} P(\hat{D}_{t'} < X_a^*)$$

$$\Delta^+ = B \sum_{t'=a}^{t_1-1} P(\hat{D}_{t'} \geq X_a^*) .$$

If  $t_1 = T+1$ ,  $\Delta^- := \Delta^- + C$ . Go to Step 4.

*Step 4* (reduce peak): If  $\Delta^- \geq \Delta^+$ , let  $X_t^* := X_t^* - 1, \forall a \leq t \leq b$ ; otherwise, let  $t_0 = b + 1$ . Go to Step 2.

*Step 5* (adjust  $\mathbf{X}^*$ ): Let  $X_t^* := \max\{X_{t-1}^*, X_t^*\}, \forall l \leq t \leq k$ . Deadhead orders in each period is  $x_t^* = X_t^* - X_{t-1}^*, \forall l \leq t \leq k$ .

So far our discussion restrict to the problem of planning for inbound deadheading. In the next section, we generalize the problem by incorporating the option of outbound deadheading.

### 3.6 MODEL 2: BI-DIRECTIONAL DEADHEADING

In this section, we discuss a model that considers both inbound and outbound deadheading to balance crew holding cost and train delay cost. For a given deadheading plan  $\mathbf{x} = (x_1, \dots, x_T)$  and  $\mathbf{y} = (y_1, \dots, y_T)$ , the expected crew holding and train delay cost under all possible train schedules  $\omega$  is as follows:

$$\begin{aligned} & \mathbb{E}_{\omega} \sum_{t=1}^T \left[ H(S_t^{\omega} + X_t - D_t^{\omega} - Y_t)^+ + B(D_t^{\omega} + Y_t - S_t^{\omega} - X_t)^+ \right] \\ &= \sum_{t=1}^T \mathbb{E} \left[ H(X_t - Y_t - \hat{D}_t)^+ + B(\hat{D}_t + Y_t - X_t)^+ \right]. \end{aligned}$$

The deadheading cost is  $\sum_{t=1}^T C(x_t + y_t)$ . The problem of finding an optimal deadheading plan can be formulated as (3.6), and is called [IOCP].

$$\begin{aligned} \text{[IOCP]} \quad & \text{minimize} \quad \sum_{t=1}^T \mathbb{E} \left[ H(X_t - Y_t - \hat{D}_t)^+ + B(\hat{D}_t + Y_t - X_t)^+ \right] + C(x_t + y_t) \\ \text{s.t.} \quad & x_t = 0, \forall t = 1, \dots, l-1 \\ & x_t \geq 0, \forall t = l, \dots, T \\ & y_t \geq 0, \forall t = 1, \dots, T \end{aligned} \tag{3.6}$$

In the following sections, we propose two methods to solve [IOCP]. Section 3.6.1 extends the recursive method of Section 3.5.1 to find the optimal solution to [IOCP]. Section 3.6.2 proposes a heuristic algorithm that utilizes the idea of newsvendor ordering policy and marginal cost analysis.

### 3.6.1 Recursive method for bi-directional deadheading

The recursive method proposed in Section 3.5.1 can be readily extended to the problem of bi-directional deadhead planning. Let  $F_t(X_t, Y_t)$  be the minimum cumulative cost up to period  $t$  (inclusive) given that  $X_t$  inbound deadheads are available and  $Y_t$  outbound deadheads are ordered on or before period  $t$ . The recursive function from period  $t - 1$  to period  $t$  is as follows:

**Recursive function:**

$$F_t(X_t, Y_t) = \min_{x_t, y_t} F_{t-1}(X_{t-1}, Y_{t-1}) + Cx_t + Cy_t + E \left[ H(X_t - Y_t - \hat{D}_t)^+ + B(\hat{D}_t + Y_t - X_t)^+ \right], \forall t = 1, \dots, T \quad (3.7)$$

The initial condition is  $F_0(X_0 = 0, Y_0 = 0) = 0$ . The optimal value function is  $\min_{X_T, Y_T} F_T(X_T, Y_T)$ .

Now we discuss the search space of  $X_t$  and  $Y_t$ . Let  $X_t^{\max}$  and  $Y_t^{\max}$  be the upper bounds of  $X_t$  and  $Y_t$ , respectively. For periods 1 to  $l - 1$ , only outbound deadheading is available, thus,  $X_t^{\max} = 0$ . Outbound deadheading is bounded by the maximum available supply, i.e.,  $Y_t^{\max} = \max\{Y_{t-1}^{\max}, S_t^{\max} - D_t^{\min}\} = \max_{t' \leq t} \{-\hat{D}_{t'}^{\min}\}$ , for  $t < l$ . Starting from period  $l$ ,  $X_t^{\max} = D_t^{\max}$  and  $Y_t^{\max} = S_t^{\max}$ . These upper bounds correspond to the worst case scenario where all regular demands are satisfied by inbound deadheading and all regular supplies are taken back to the home station by outbound deadheading. We can reduce the search space of  $Y_t$  by finding its upper bound for any given value of  $X_t$ , denoted by  $Y_t^{\max}(X_t)$ . For given  $X_t$ , let  $\chi_t = \min\{X_t, X_{t-1}^{\max}\}$ . Then

$Y_t^{\max}(X_t) = \max\{Y_{t-1}^{\max}(\chi_t), X_t + S_t^{\max} - D_t^{\min}\} = \max\{Y_{t-1}^{\max}(\chi_t), X_t - \hat{D}_t^{\min}\}$ . We use  $Y_t^{\max}$  or  $Y_t^{\min}(X_t)$ , whichever is smaller, as the upper bound of  $Y_t$ . Lemma 3.2 shows that we will not simultaneously have inbound and outbound deadheads in the same period in any optimal solution. Therefore, for a given pair of  $X_t$  and  $Y_t$ , we only need to search  $x_t = 0, \dots, X_t$ ,  $y_t = 0$  and  $x_t = 0, y_t = 1, \dots, Y_t$ . The computational complexity of this recursive method is  $O(TD_T^{\max}S_T^{\max}(D_T^{\max} + S_T^{\max}))$ .

**Lemma 3.2:**

For any optimal solution to [IOCP],  $x_t^* y_t^* = 0, \forall t$ .

**Proof:** Suppose  $(\mathbf{x}^*, \mathbf{y}^*)$  is an optimal solution to [IOCP], and there exists a period  $k$  with both inbound and outbound deadheads, i.e.,  $x_k^* y_k^* > 0$ . Let  $x'_k = x_k^* - \min\{x_k^*, y_k^*\}$  and  $y'_k = y_k^* - \min\{x_k^*, y_k^*\}$ . Let  $\mathbf{x}' = [x_1^*, \dots, x'_k, \dots, x_T^*]$  and  $\mathbf{y}' = [y_1^*, \dots, y'_k, \dots, y_T^*]$ . It is easy to see that  $Z'_t = \sum_{i=1}^t [\mathbf{x}' - \mathbf{y}']_i = \sum_{i=1}^t [\mathbf{x}^* - \mathbf{y}^*]_i = Z_t$ ,  $\forall t = 1, \dots, T$ . Therefore, solution  $(\mathbf{x}', \mathbf{y}')$  has the same expected crew holding and train delay cost as  $(\mathbf{x}^*, \mathbf{y}^*)$  but lower deadheading cost. Hence,  $(\mathbf{x}^*, \mathbf{y}^*)$  is not optimal. ♦

### 3.6.2 Newsvendor-based heuristic for bi-directional deadheading

To derive the solution, we define  $z_t = x_t - y_t$  as the net deadheading quantity in period  $t$ , for  $t = 1, \dots, T$ . Accordingly, let  $Z_t = \sum_{i=1}^t z_i$ . Note that  $z_t \leq 0$  for  $t = 1, \dots, l-1$  because  $x_t = 0$  and  $y_t \geq 0$ . From Lemma 3.2, we can show that  $x_t + y_t = |x_t - y_t| = |z_t|$ . Therefore, we can re-write [IOCP] as follows:



$$\begin{aligned}
& \text{minimize} \quad \sum_{t=1}^T \mathbb{E} \left[ H(Z_t - \hat{D}_t)^+ + B(\hat{D}_t - Z_t)^+ \right] + C|z_t| \\
& \text{s.t.} \quad z_t \leq 0, \forall t = 1, \dots, l-1
\end{aligned} \tag{3.8}$$

We temporarily assume that the variables  $Z_t$  and  $\hat{D}_t$  are continuous, and  $\Phi_t$  has continuous first-order derivatives. The objective function of (3.8) is not differentiable at  $z_t = 0$ , thus, the KKT conditions do not apply. However, if we ignore the deadheading cost and relax the constraint in (3.8), the problem is simplified to the unconstrained minimization problem of (3.9):

$$\text{minimize} \quad F(\mathbf{Z}) = \sum_{t=1}^T \mathbb{E} \left[ H(Z_t - \hat{D}_t)^+ + B(\hat{D}_t - Z_t)^+ \right] \tag{3.9}$$

By setting the derivatives  $dF(\mathbf{Z})/dz_t = 0, \forall t = 1, \dots, T$ , we can show that the optimal solution to (3.9) is  $Z_t^* = \Phi_t^{-1}(\rho)$ , where  $\rho = B/(H+B)$ . Now since supply and demands are discrete in our problem, we approximate the solution to (3.9) using  $Z_t^* = \lceil \Phi_t^{-1}(\rho) \rceil, \forall t = 1, \dots, T$ , where  $Z_t^*$  is the smallest integer such that  $P(\hat{D}_t \leq Z_t^*) \geq \rho$ . Then we can calculate the net deadheading quantity in each period using  $z_t^* = Z_t^* - Z_{t-1}^*$ . Note that  $z_t^*$  could be positive, negative, or zero because the sequence of  $Z_t^*$  may be neither monotonically increasing nor decreasing.

After obtaining the initial solution, we apply two types of local search using the idea of marginal cost analysis. The first local search procedure seeks for the opportunity to reduce cost by cancelling out some pairs of positive  $z_t^*$  (i.e., inbound deadheads) and negative  $z_t^*$  (i.e., outbound deadheads). Suppose  $(z_a^*, z_b^*)$  is a pair of

inbound and outbound deadheads such that  $z_a^* z_b^* < 0$  and  $z_t^* = 0, \forall a < t < b$ . If  $z_a^* > 0, z_b^* < 0$ , then cancelling out one pair of deadheads saves deadheading cost for two persons plus crew holding cost from periods  $a$  to  $b - 1$ . The marginal cost is the increased train delay cost during the same periods. Note that cancelling deadheads in periods  $a$  and  $b$  will not affect costs for periods  $t < a$  and  $t \geq b$  because  $Z_t$  remain unchanged for these periods. If the marginal benefit is higher than marginal cost, we can obtain a better solution by cancelling out a pair of deadheads. If  $z_a^* < 0, z_b^* > 0$ , the same analysis apply. The only difference is that marginal benefit equals the saving in deadheading cost plus reduced train delay cost, while marginal cost equals the extra crew holding cost from periods  $a$  to  $b - 1$ .

The second local search procedure attempts to cancel some deadheads towards the end of planning horizon. Suppose there exists period  $a$  such that  $z_a^* > 0$  and  $z_t^* = 0, \forall t > a$ . Then reduce  $z_a^*$  by one saves 1 person's trip cost and potential crew holding cost for all following periods starting from  $a$ . The marginal cost, though, would be the expected increasing in train delay cost. If marginal benefit outweighs the marginal cost increase, we will reduce  $z_a^*$  by one for a lower-cost solution. Similarly, if  $z_a^* < 0$  and  $z_t^* = 0, \forall t > a$ , increase  $z_a^*$  by one saves deadheading cost and train delay cost, but incurs more crew holding cost. Cancelling one outbound deadhead is beneficial when the marginal saving is larger than the marginal increase in cost.

We apply the two local search procedures sequentially until no more saving can be achieved by cancelling more deadheads. However, the resulting solution is not

necessary feasible because we may have some inbound deadheads for periods before  $l$ .

In such case, we adjust the sequence of  $Z_t^*$  by setting

$Z_t^* := \min\{Z_{t-1}^*, Z_t^*\}, \forall t = 1, \dots, l-1$ . Essentially, this adjustment either cancels out

inbound deadheads with following outbound deadheads in periods before  $l$  or delays the

inbound deadheads till period  $l$ . We describe the details of this heuristic below:

### Heuristic 3: relax\_and\_cancel:

*Step 1:* Compute initial solution  $Z_t^* = \lceil \Phi_t^{-1}(\rho) \rceil$ ,  $z_1^* = Z_1^*$ , and  $z_t^* = Z_t^* - Z_{t-1}^*$ ,  $\forall t = 1, \dots, T$ . Let  $t_0 := 1$  and  $a := t_0$ .

*Step 2:* If there exists  $b > a$  such that  $z_a^* z_b^* < 0$  and  $z_t^* = 0, \forall a < t < b$ , go to Step 3; otherwise, let  $a := a+1$ . If  $a < T$ , repeat Step 2; otherwise  $a = T$ , go to Step 4.

*Step 3:* Compute the marginal benefit,  $\Delta^-$ , and cost,  $\Delta^+$ , for canceling one deadhead in period  $a$  and one in period  $b$ .

Case 1: if  $z_a^* > 0$  and  $z_b^* < 0$ :

$$\Delta^- = 2C + \sum_{t=a}^{b-1} H\Phi_t(Z_a^* - 1)$$

$$\Delta^+ = \sum_{t=a}^{b-1} B[1 - \Phi_t(Z_a^* - 1)]$$

Case 2: if  $z_a^* < 0$  and  $z_b^* > 0$ :

$$\Delta^- = 2C + \sum_{t=a}^{b-1} B[1 - \Phi_t(Z_a^*)]$$

$$\Delta^+ = \sum_{t=a}^{b-1} H\Phi_t(Z_a^*)$$

If  $\Delta^- \geq \Delta^+$ , let  $z_a^* := (|z_a^*| - 1) \cdot z_a^* / |z_a^*|$  and  $z_b^* := (|z_b^*| - 1) \cdot z_b^* / |z_b^*|$ . Update

$Z_t^* := Z_{a-1}^* + z_a^*, \forall t = a, \dots, b-1$ . Set  $a := t_0$ . Else if  $\Delta^- < \Delta^+$ , do not cancel the deadheads. Set  $t_0 := b$  and  $a := t_0$ . Go to Step 2.

*Step 4:* If  $z_a^* \neq 0$ , go to Step 5; otherwise, let  $a := a - 1$ . If  $a \geq 1$ , repeat Step 4; otherwise, go to Step 6.

*Step 5:* Compute marginal benefit and cost if we cancel one deadhead in period  $a$ .

Case 1: if  $z_a^* > 0$ :

$$\Delta^- = C + \sum_{t=a}^T H\Phi_t(Z_a^* - 1)$$

$$\Delta^+ = \sum_{t=a}^T B[1 - \Phi_t(Z_a^* - 1)]$$

Case 2: if  $z_a^* < 0$ :

$$\Delta^- = C + \sum_{t=a}^T B[1 - \Phi_t(Z_a^*)]$$

$$\Delta^+ = \sum_{t=a}^T H\Phi_t(Z_a^*)$$

If  $\Delta^- \geq \Delta^+$ , let  $z_a^* := (|z_a^*| - 1) \cdot z_a^* / |z_a^*|$ . Update  $Z_t^* := Z_{a-1}^* + z_a^*, \forall t = a, \dots, T$ . Go

to Step 4. Else if  $\Delta^- < \Delta^+$ , do not cancel deadhead in period  $a$ . Go to Step 6.

*Step 6:* Let  $Z_0^* = 0$  and  $Z_t^* := \min\{Z_{t-1}^*, Z_t^*\}, \forall t = 1, \dots, l-1$ . Compute deadheading plan  $(\mathbf{x}^*, \mathbf{y}^*)$ :  $z_t^* = Z_t^* - Z_{t-1}^*$ . If  $z_t^* > 0$ , then  $x_t^* = z_t^*$  and  $y_t^* = 0$ ; if  $z_t^* < 0$ , then  $x_t^* = 0$  and  $y_t^* = -z_t^*$ ; if  $z_t^* = 0$ , then  $x_t^* = 0$  and  $y_t^* = 0$ ,  $\forall t = 1, \dots, T$ .

This heuristic can also be applied to the problem with only inbound deadheading options with some minor changes. First, when we compute the initial solution in Step 1, we only need to compute  $Z_t^*$  for  $t \geq l$  and let  $Z_t^* = 0, \forall t < l$ . Steps 2 to 5 remain unchanged. In Step 6, instead of eliminating inbound deadheads in periods before  $l$ , we need to eliminate all outbound deadheads, that is, let  $Z_t^* := \max\{Z_{t-1}^*, Z_t^*\}, \forall t = l, \dots, T$ . The inbound deadheading plan  $\mathbf{x}^*$  is  $x_t^* = Z_t^* - Z_{t-1}^*, \forall t$ . In Section 3.7, we will compare this modified relax\_and\_cancel method against the two heuristics, base\_NV and reduce\_peak, that are designed for [IBCP].

### 3.7 COMPUTATIONAL EXPERIMENTS

In this section, we generate random instances to test the performance of the

heuristic methods under various settings of parameters. In Sections 3.7.1 to 3.7.4, we discuss the details of the testing instances, including the distribution of base train schedules, deviation of actual schedule from base schedule, and the distribution of cumulative supplies and demands. Then, starting from Section 3.7.5, we compare the performance of the heuristics against the optimal solution. We also show the sensitivity of the heuristics' performance to various cost parameters and the variance in train schedules. Finally, we conclude with observations and insights from the testing results.

### **3.7.1 Distribution of base train schedule**

Base schedule refers to the scheduled arrival time (inbound trains) and departure time (outbound trains) at the beginning of the planning horizon. For inbound trains, the base schedule is the trains' scheduled arrival time plus the amount of time needed for a crew member to get rested, i.e., the scheduled rested time. For a given planning horizon of periods 1 to  $T$ , we generate  $|I|$  inbound trains and  $|J|$  outbound trains. The number of trains,  $|I|$  and  $|J|$ , are randomly selected in a given range (say, 20 to 60 trains in each direction in a 36-hour horizon). For each inbound and outbound train, its base schedule is generated according to certain distribution on  $[0, T\tau]$ , where  $\tau$  is period length. Here we consider three distributions: (1) Uniform distribution, (2) High-Low distribution where the probability of train arriving/departing in the first half of the horizon is twice as high as the probability of it falling in the second half of the horizon, and (3) Low-High distribution where the probability of train arriving/departing in the second half of the horizon is twice as high as the probability of it falling in the first half of the horizon. Figure 3.1 depicts these three distributions.

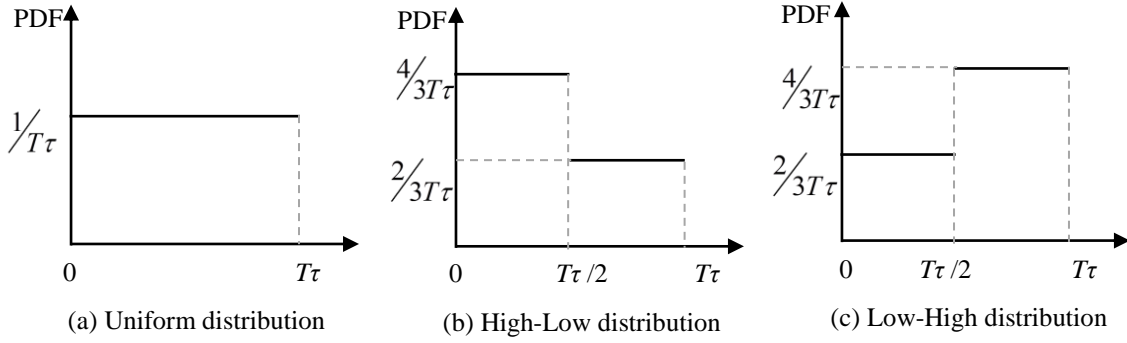


Figure 3.1 Distribution of base train schedule

### 3.7.2 Train schedule deviation

The actual arrival/departure time of train will deviate from its base schedule because of the randomness in its dwell time in upstream stations and traversal time between stations. In the literature, several distributions, such as exponential, gamma, and Weibull distributions were used to model the delay of trains. See Krüger et al. (2013) for a discussion of different train delay models. In our testing, we use shifted gamma distributions to model the difference of actual arrival/departure time compared to the base schedule. Let  $\text{Gamma}(\alpha, \beta, \theta)$  denote a shifted gamma distribution, where  $\alpha$  is the shape parameter,  $\beta$  is the scale parameter, and  $\theta$  is the shift. The mean and standard deviation are  $\alpha\beta + \theta$  and  $\sqrt{\alpha\beta}$ , respectively. We use two different sets of parameters for trains scheduled at different times in the horizon. If train's scheduled (base) arrival time (inbound train) or departure time (outbound train) is within 8 hours from the start of planning horizon, let the mean be 0.5 hours, standard deviation be 0.6 hours, and shift be -0.5 hours. For later trains, we use mean of 2 hours, standard

deviation of 1.2 hours, and shift of -1 hour. The two corresponding distributions are  $\text{Gamma}(2.78, 0.36, -0.5)$  and  $\text{Gamma}(6.25, 0.48, -1)$ . Figure 3.2 plots the PDF's of these two distributions.

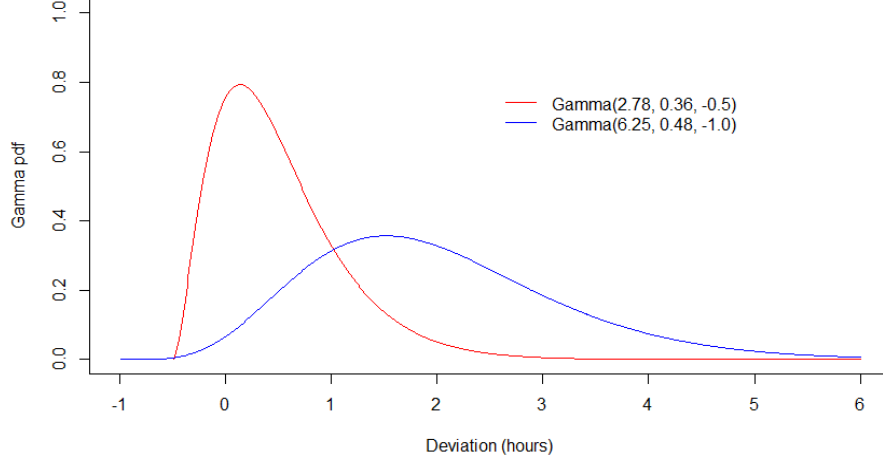


Figure 3.2 Two Gamma distributions of train schedule deviation

Next we discretize the gamma distributions to compute the probability of deviating  $k$  periods from base schedule. Let  $\delta$  be the deviation in schedule. The probability of deviating  $k$  periods is  $P(\delta \leq (k+0.5)\tau) - P(\delta \leq (k-0.5)\tau)$ . For an inbound train  $i$ , let  $t_i^b$  be the base scheduled period (rested period) and  $t_i^a$  be the actual realization. Let  $p_{it}$  be the probability of  $t_i^a = t$ ,  $G_{\alpha,\beta,\theta}$  be the CDF of  $\text{Gamma}(\alpha, \beta, \theta)$ . We can compute  $p_{it}$  as follows:

$$\begin{aligned} p_{it} &= p(t_i^a = t) = p(t_i^a - t_i^b = t - t_i^b) = p(\delta = t - t_i^b) \\ &= G_{\alpha,\beta,\theta}((t - t_i^b + 0.5)\tau) - G_{\alpha,\beta,\theta}((t - t_i^b - 0.5)\tau), \forall i \in I, t = 1, \dots, T \end{aligned}$$

The distribution of outbound train's schedule,  $p_{jt}$ , can be computed using the same

method.

Now we have obtained the distributions of each inbound and outbound train's arrival/departure time. However, our algorithms do not use the distributions of trains' schedule directly. Instead, we need the distributions of cumulative supplies, demands, and net demands. These random variables are closely related to the distribution of train schedule. In the next section, we discuss methods to compute the distributions of  $S_t$ ,  $D_t$ , and  $\hat{D}_t$  using  $p_{it}$  and  $p_{jt}$ .

### 3.7.3 Distribution of cumulative supplies and demands

Given the distributions of each train's schedule, we are now ready to compute the distributions of cumulative supplies, demands, and net demands. Let  $I$  and  $J$  be the sets of inbound and outbound trains considered in the planning horizon which are index by  $i = 1, 2, \dots, |I|$  and  $j = 1, \dots, |J|$ , respectively. We introduce indicator  $u_{it} = 1$  if the crew on inbound train  $i$  is rested (and thus a supply) on or before  $t$ , and  $v_{jt} = 1$  if outbound train  $j$  is ready to depart (and thus a demand) on or before  $t$ . The probabilities of  $u_{it} = 1$  and  $v_{jt} = 1$  are  $\sum_{t'=1}^t p_{it'}$  and  $\sum_{t'=1}^t p_{jt'}$ , respectively. For a given period  $t$ , the cumulative supply and demand up to period  $t$  are  $S_t = \sum_{i \in I} u_{it}$  and  $D_t = \sum_{j \in J} v_{jt}$ . The distributions of  $S_t$  and  $D_t$  are Poisson Binomial distribution because  $u_{it}$  and  $v_{jt}$  are independent and non-identical Bernoulli random variables. Researchers have started to study Poisson Binomial distribution since a long time ago. Both approximation and exact algorithms have been proposed to calculate its probability mass function (PMF) and



CDF (e.g., see Hong (2013)). Here we use a recursive function to compute the PMF of  $S_t$ . This method is proposed by Barlow and Heidtmann (1984), and reviewed in Hong (2013). Let  $S_t^n = \sum_{i=1}^n u_{it}$  and  $\xi_t(n, k) = \Pr(S_t^n = k)$ , for  $0 \leq n \leq |I|$ . Note that  $S_t^{[I]} = S_t$ . Equation (3.10) computes  $\xi_t(n, k)$  recursively for all  $k \leq n$ .

$$\xi_t(n, k) = \left( \sum_{t'=1}^t p_{nt'} \right) \xi_t(n-1, k-1) + \left( 1 - \sum_{t'=1}^t p_{nt'} \right) \xi_t(n-1, k), \forall 0 \leq k \leq n \leq |I| \quad (3.10)$$

The boundary conditions are  $\xi_t(0, 0) = 1$ ,  $\xi_t(n, -1) = 0$ , and  $\xi_t(n, n+1) = 0$  for  $n = 0, \dots, |I| - 1$ . The PMF of  $S_t$  is  $\Pr(S_t = k) = \xi_t(|I|, k)$ . The same computation applies to  $D_t$  if we define  $D_t^n = \sum_{j=1}^n v_{jt}$  and  $\xi_t(n, k) = \Pr(D_t^n = k)$ , for  $0 \leq n \leq |J|$ . The PMF of  $D_t$  is  $\Pr(D_t = k) = \xi_t(|J|, k)$ . We can reduce the computation time of this method by setting  $S_t = \sum_{i \in I_t} u_{it}$  and  $D_t = \sum_{j \in J_t} v_{jt}$ , where  $I_t$  and  $J_t$  are subsets of  $I$  and  $J$  with  $I_t = \{i \in I : \sum_{t'=1}^t p_{it'} > 0\}$  and  $J_t = \{j \in J : \sum_{t'=1}^t p_{jt'} > 0\}$ .

Given the distributions of  $S_t$  and  $D_t$ , it is straight forward to compute the PMF of net demands  $\hat{D}_t = D_t - S_t$  using equation (3.11):

$$\Pr[D_t - S_t = k] = \sum_{a=0}^{|I|} \Pr(S_t = a) \Pr(D_t = a + k), \forall -|I| \leq k \leq |J| \quad (3.11)$$

However, if we are only interested in the PMF of  $\hat{D}_t$  but not that of  $S_t$  and  $D_t$ , then we can compute it directly using equation (3.10) because  $\hat{D}_t$  also follows Poisson Binomial distribution. To see this, we re-write  $\hat{D}_t$  as follows:

$$\begin{aligned}
\hat{D}_t &= D_t - S_t = \sum_{j \in J} v_{jt} - \sum_{i \in I} u_{it} \\
&= -|I| + \sum_{j \in J} v_{jt} + \sum_{i \in I} (1 - u_{it}) \\
&= -|I| + R_t
\end{aligned}$$

where  $-|I|$  is a constant and  $R_t = \sum_{j \in J} v_{jt} + \sum_{i \in I} (1 - u_{it})$  is a Poisson Binomial random variable. To use equation (3.10), we define  $IJ = I \cup J$ , and index it by  $m = 1, \dots, |I| + |J|$ . Let  $r_{mt} = 1$  if train  $m$  is a supply/demand on or before period  $t$ .  $\Pr(r_{mt} = 1) = \sum_{t'=1}^t p_{mt'}$  if train  $m$  is a demand and  $\Pr(r_{mt} = 1) = 1 - \sum_{t'=1}^t p_{mt'}$  if train  $m$  is a supply. Then we re-write  $R_t = \sum_{m \in IJ} r_{mt}$ , and define  $R_t^n = \sum_{m=1}^n r_{mt}$  and  $\xi_t(n, k) = \Pr(R_t^n = k)$ , for  $0 \leq n \leq |IJ|$ . Then we can apply equation (3.10) to get  $\Pr(R_t = k), \forall -|I| \leq k \leq |J|$ . The PMF of  $\hat{D}_t$  is  $\Pr(\hat{D}_t = k) = \Pr(R_t = k + |I|)$ . The CDF of  $\hat{D}_t$  is  $\Phi_t(a) = \sum_{k \leq a} \Pr(\hat{D}_t = k)$ .

### 3.7.4 Testing instances

For the computational experiments, we consider instances of 36-hour planning horizon and 5-minute periods. We generate 30 instances for each of the five settings of base schedule distribution listed in Table 3.1. The numbers of scheduled inbound/outbound trains during the planning horizon are generated randomly from a Uniform distribution in the range of [20, 60]. After generating the base schedule of each train, the distribution of their schedule deviation is generated using the two gamma distributions described in Section 3.7.2. Assume the fastest deadheading mode takes 4 hours to transit from the home station to the away station, and crew members need 10

hours to get fully rested. Then, the earliest time when an inbound deadheaded crew becomes available is after hour 14, or in period 169 (i.e.,  $l = 169$ ). On the other hand, outbound deadheading option is available in every period starting from period one if this option is considered (i.e., in model [IOCP]. The deadheading cost is 200 per person, delay cost is 500 per hour, and crew holding cost is 20 per hour. Therefore,  $C = 200$ ,  $B = 41.67$ , and  $H = 1.67$ .

Instance Group	Distribution of inbound trains	Distribution of outbound trains
<b>A</b>	Uniform	Uniform
<b>B</b>	High-Low	High-Low
<b>C</b>	High-Low	Low-High
<b>D</b>	Low-High	High-Low
<b>E</b>	Low-High	Low-High

Table 3.1 Testing groups and base schedule distributions

### 3.7.5 Compare solution methods of model [IBCP]

For model [IBCP], the recursive method described in Section 3.5.1 provides an optimal solution. We consider three heuristics that are based on the newsvendor ordering policy: (i) base\_NV (Section 3.5.2.2), (ii) reduce\_peak (Section 3.5.2.3), and (iii) relax\_and\_cancel (Section 3.6.2; modified for application on model [IBCP]). We test these different solution methods using 150 instances generated as described in the last section. The performance of these heuristic methods are measured by a number of metrics, including percentage gap of total cost and each cost component (holding cost, delay cost, and deadheading cost), gap of deadhead counts, difference in the average time of deadheading and average supply level. The percentage cost gaps are the delta costs

(i.e., heuristic solution's cost component minus optimal solution's cost component) expressed as a percentage of the optimal solution's total cost. We define average deadheading time as  $\sum_{t=l}^T tx_t / X_T$  if  $X_T > 0$ , and *null* otherwise. Average supply level is defined as  $\bar{P} := \sum_{t=1}^T P(S_t + X_t \geq D_t) / T$ . This metric shows the average risk of delaying a train due to crew shortage. The gaps of deadhead counts, average deadheading time, and average supply level are taken as the delta value using heuristic solution's value minus optimal solution's value.

In the following sections, we first compare the performance of the heuristics under the default parameter setting (see Section 3.7.4). Then we test performance sensitivity with respect to different parameter settings.

#### ***3.7.5.1 Performance comparison with default parameters***

In this section, we test the exact recursive method and three heuristic methods on 150 random instances from 5 groups. Table 3.2 shows the minimum, maximum, and average percentage gap of total cost with respect to the optimal cost. Recall that the total cost includes expected crew holding cost and train delay cost, and total deadheading cost from period  $l$  to  $T$ . First, we notice that the minimum gaps of all three methods are zero for all testing groups, meaning that all three heuristics were able to find the optimal solution for some instances. Second, `reduce_peak` method has the best performance among the three heuristics with an overall average gap of only 0.18%. In particular, this heuristic obtained optimal solutions for all instances in groups C, D, and E. Method `base_NV` has the worst overall performance except for group C (`relax_and_cancel` has the

worst performance for this group). Especially, the maximum gap is as high as 46.4% in group A. Finally, we can see that the effectiveness of the two adjusted newsvendor-ordering heuristics is dependent on the distribution of trains. In group A, `reduce_peak` and `relax_and_cancel` reduced the maximum gap from 46.4% to 7.7% and 4.5%, respectively. However, both methods failed to improve the `base_NV` solution for the worst cases in group B.

Instance group	Avg. non-monotonicity	Base_NV			Reduce_peak			Relax_and_cancel		
		Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.
A	1.63	0.0%	46.4%	4.5%	0.0%	7.7%	0.3%	0.0%	4.5%	0.4%
B	0.90	0.0%	17.5%	1.3%	0.0%	17.5%	0.6%	0.0%	17.3%	1.3%
C	0.20	0.0%	3.0%	0.2%	0.0%	0.0%	0.0%	0.0%	4.2%	0.3%
D	1.10	0.0%	16.7%	1.6%	0.0%	0.0%	0.0%	0.0%	5.3%	0.5%
E	1.03	0.0%	15.2%	1.4%	0.0%	0.0%	0.0%	0.0%	6.9%	1.2%

Table 3.2 Three heuristics for [IBCP]: percentage gap of total cost

Next we compare the cost components. Figure 3.3 and Figure 3.4 display the percentage gap of crew holding cost and train delay cost, respectively, for each instance. The x-axis is instance name, sorted from left to right in the increasing order of `base_NV`'s total cost gap. These two pictures show that `base_NV` and `reduce_peak` always have non-negative gap of crew holding cost and non-positive gap of train delay cost, indicating that these two methods tend to be more conservative than the optimal solution and keep a higher level of crew inventory. This observation is consistent with the fact that we round up  $X_l^*$  to be the smallest integer with  $P(\hat{D}_l \leq X_l^*) \geq \rho, \forall l \leq t \leq k-1$ . Figure 3.5 better illustrates this observation: the average supply level is always higher in the

solutions from base\_NV and reduce\_peak. On the other hand, method relax\_and\_cancel may have positive, negative, or zero gap of all costs. In fact, in most of these instances, this method finds solutions with lower crew holding cost and higher train delay cost. The second observation is that the performance of base\_NV and reduce\_peak is correlated in some sense. As we sort the instances in increasing order of base\_NV's total cost gap in these plots, the gaps are apparently higher at the right end of these plots for both base\_NV and reduce\_peak. Further, the gaps of reduce\_peak are never larger than those of base\_NV. By design, reduce\_peak is based on the base\_NV method, and it will revise the deadheading plan only if the revision produces a better solution.

Next we analyze the solutions from the perspective of deadhead counts and average deadheading time. Figure 3.6, we plot the gap of average deadheading time (y-axis) verse the delta deadheading count (x-axis). An outlier from the series of relax\_and\_cancel was removed for a better presentation of the figure. First, notice that there are no blue or red markers on the left side of y-axis. This observation shows that the base\_NV and reduce\_peak methods always deadhead the same number of or more crew members, and so solutions of these methods always have the same or higher level of supply. In addition, all the blue and red marks on the y-axis lie on the non-positive half of the y-axis, indicating that base\_NV and reduce\_peak methods always deadhead at the same time or earlier than the optimal solution if they have the same deadhead count. As for relax\_and\_cancel, it can deadhead more, the same, or less than the optimal solution. Even if it has the same deadhead count as the optimal solution, the time of deadheading may be earlier or later than the optimal deadheading plan.

Finally, let us get some insight of why the performance of heuristics, especially `base_NV`, varies widely across instances. Recall that we made several assumptions to derive the solution in Section 3.5.2.1. The most restrictive assumption is that the sequence of net cumulative demands is stochastically increasing. To approximately measure the extent to which an instance violates this assumption, let  $\mathbf{X}^*$  be the sequence of cumulative ordering quantity from Step 5 of `base_NV`, i.e.,  $X_t^* = \left[ \left[ \Phi_t^{-1}(\rho) \right] \right]^+$ ,  $\forall l \leq t \leq k-1$ , and  $X_k^* = \left[ Q^* \right]^+$ . Define *degree of non-monotonicity* as  $\sum_{t=l}^{k-1} \left[ X_t^* - X_{t+1}^* \right]^+$ . Figure 3.7 plots the percentage gap of the total cost of `base_NV` solutions and the non-monotonicity of each testing instance. Clearly, there is a strong correlation between the performance of `base_NV` and non-monotonicity: `base_NV` tends to have better performance when the sequence of  $\mathbf{X}^*$  is closer to monotonically increasing. Moreover, in all instances with non-decreasing  $\mathbf{X}^*$  (i.e., non-monotonicity = 0), the `base_NV` method obtains an optimal solution (i.e., total cost gap = 0). From a different point of view, Figure 3.8 plots the total cost gap of all three heuristic while the instances are sorted from left to right with increasing degree of non-monotonicity. This figure shows that `base_NV` has good performance when the sequence of  $\mathbf{X}^*$  has low degree of non-monotonicity. When  $\mathbf{X}^*$  have many peaks and valleys, `base_NV` may result in solutions with large gap. See Table 3.2 for the relationship between the average degrees of non-monotonicity and the average total cost gap of `base_NV`. However, we can effectively improve the solution by reducing peaks of  $\mathbf{X}^*$  as described in method `reduce_peak`. On

the other hand, relax\_and\_cancel heuristic provides good solutions and the solution quality is less dependent on the monotonicity of  $\mathbf{X}^*$ .

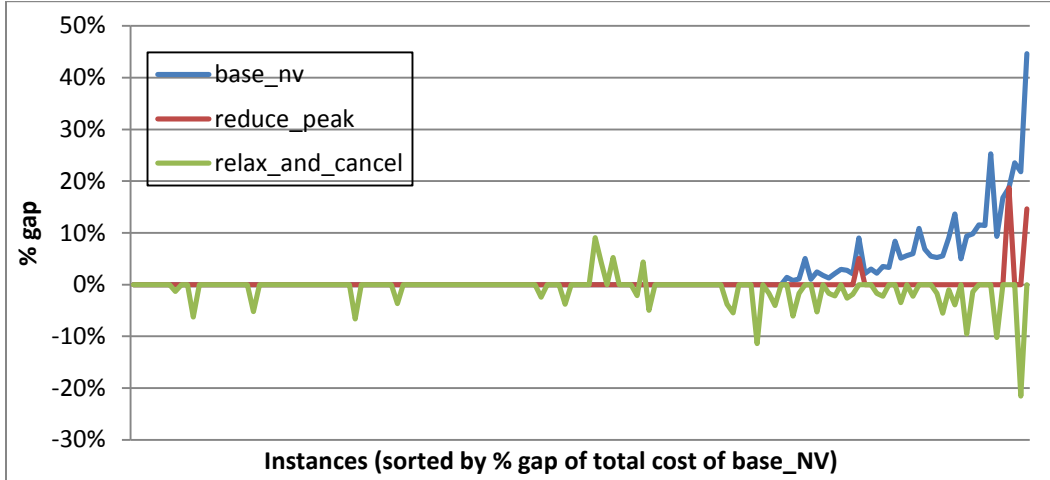


Figure 3.3 Gap of expected crew holding cost

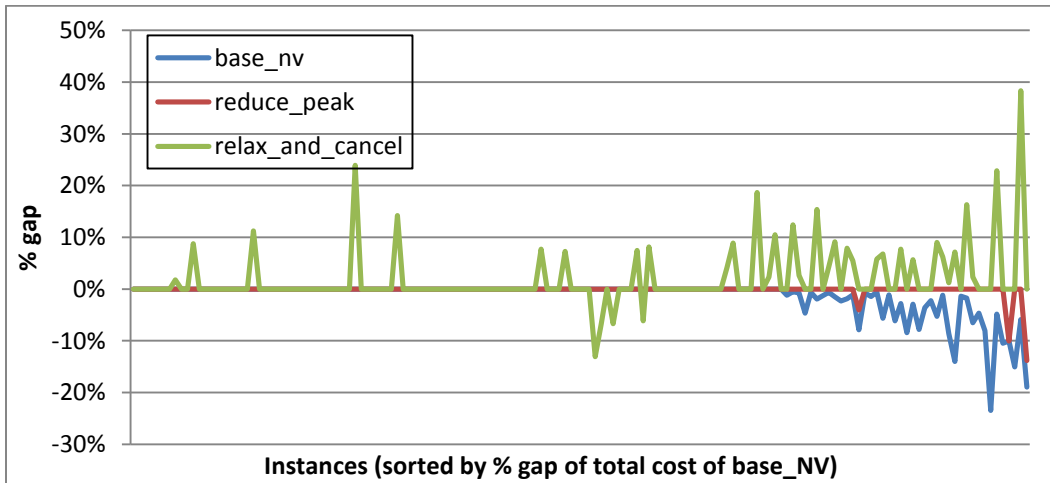


Figure 3.4 Gap of expected train delay cost



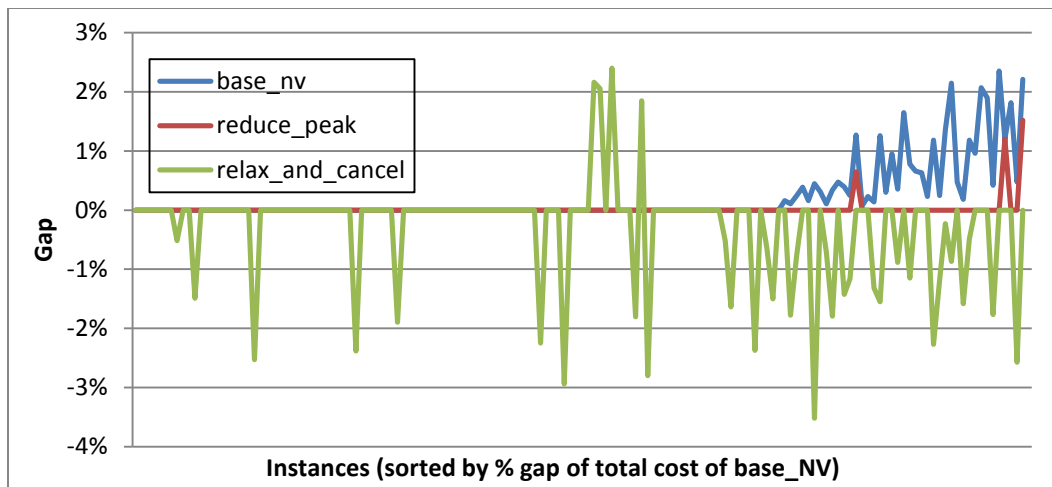


Figure 3.5 Gap of average supply level

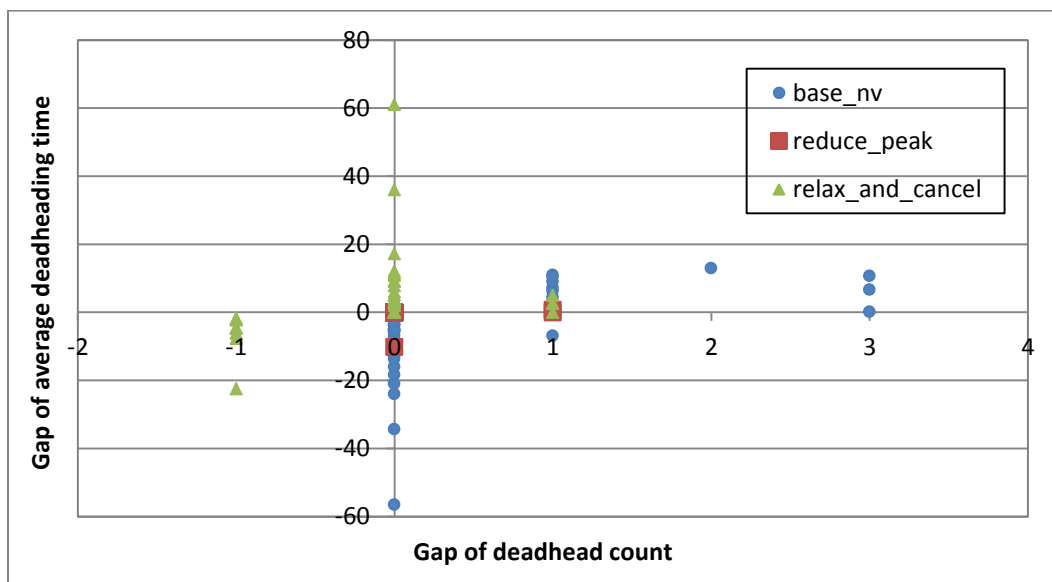


Figure 3.6 Gaps of deadhead count and average deadheading time

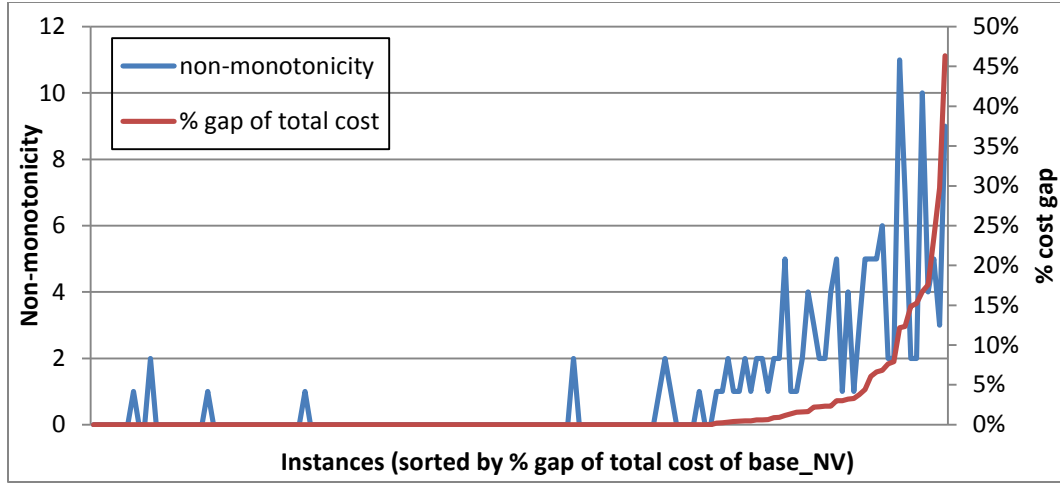


Figure 3.7 Base\_NV: percentage gap of total cost vs non-monotonicity of  $X^*$

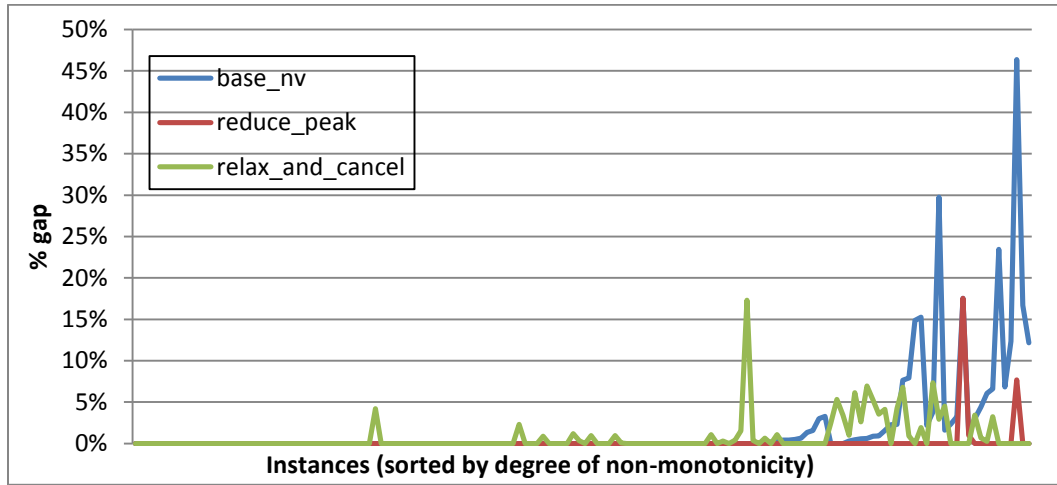


Figure 3.8 Percentage gap of total cost

### 3.7.5.2 Sensitivity to cost parameters

In this section, we examine the sensitivity of the heuristics' performance to different cost parameters. Crew holding cost  $H$  and train delay cost  $B$  impact the heuristic solutions in two ways. First, they will change the critical ratio  $\rho = B/(H + B)$ .

Second, they impact the marginal cost analysis when we adjust the deadheading plan in `reduce_peak` and `relax_and_cancel`. Here we perform three experiments, each change one parameter of  $H$ ,  $B$ ,  $C$  and keep the other two at their default values. We will compare four metrics: percentage gap of three heuristics' total cost with respect to the optimal objective value, expected crew holding time (number of held crew period) and train delay time (number of delayed train period), and number of deadheads. All metrics were averaged across 150 testing instances.

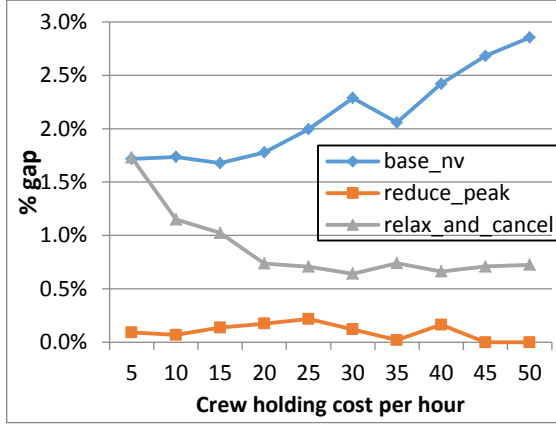
In the first experiment, we vary the crew holding cost from 5 to 50 per hour. Figure 3.9 presents the changes of all four metrics. Figure 3.9 (a) shows that `base_NV`'s performance deteriorates as  $H$  increases. `Relax_and_cancel` significantly improves performance when holding cost increases from 5 to 20 per hour, and then steady keeps around 0.7%. `Reduce_peak` performed best among all three heuristics and its gap is relatively insensitive to the change of  $H$ . Figure 3.9 (b) to (c) shows that all methods had less crew holding time, longer train delay time, and fewer inbound deadheads as we increase  $H$ , which is as expected. However, the rate of change varies for different methods, hence, the trends in the change of total cost gap are different.

As noted in the last section, `base_NV` tends to order more deadheads than the optimal solution. As  $B$  is significantly larger than  $H$ ,  $\rho$  does not reduce much as we increase  $H$ , and thus `base_NV` does not reduce deadheads as much as the optimal solution does. As Figure 3.9 (d) shows, the gap between the curves of `base_NV` and optima becomes larger when  $H$  increases, i.e., `base_NV` over-orders more. Because over-

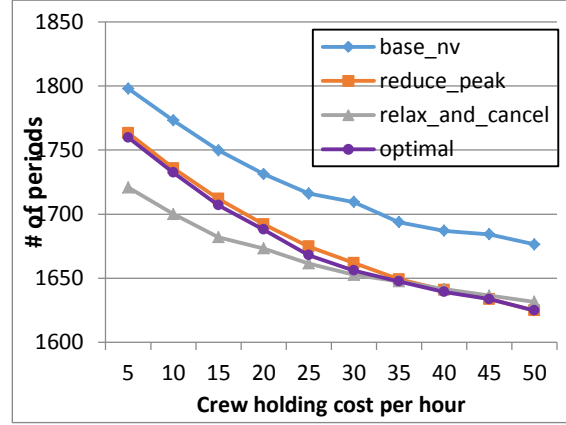
ordering is more costly with larger  $H$ , it is expected that base\_NV's overall performance becomes worse when holding crew is expensive.

For relax\_and\_cancel, its performance mainly depends on whether it cancels the right pairs of inbound and outbound deadheads. For example, suppose relax\_and\_cancel starts with a solution with an inbound deadhead in period  $t_1$ , an outbound deadhead in period  $t_2 > t_1$ , and another inbound deadhead in period  $t_3 > t_2$ . By design, the algorithm will cancel out the first inbound deadhead with the outbound deadhead as long as it is beneficial to do so (i.e., total cost reduces). However, the optimal solution might keep the inbound deadhead in  $t_1$  and cancel the pair of deadheads in periods  $t_2$  and  $t_3$ . When  $H$  is very small,  $\rho$  is very close to 1 ( $\rho = 0.99$  when holding cost is 5 per hour). The initial solution of this algorithm tends to have more inbound and outbound deadheads when  $\rho$  is large. In this case, it is more likely that the algorithm will end with canceling out the wrong pairs of deadheads, resulting in a higher gap in total cost compared to the optimal solution.

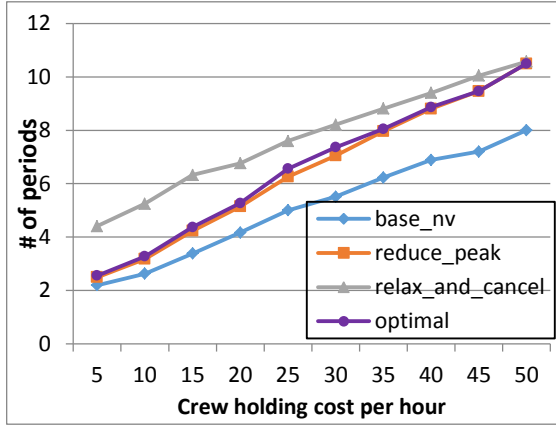
For reduce\_peak, its curves are very close to those for the optimal solution in Figure 3.9 (b), (c), and (d), which suggests that reduce\_peak generates solutions that are very similar (or the same) to the optimal solutions. Hence, the overall performance of reduce\_peak is very promising and it is robust under different values of  $H$ .



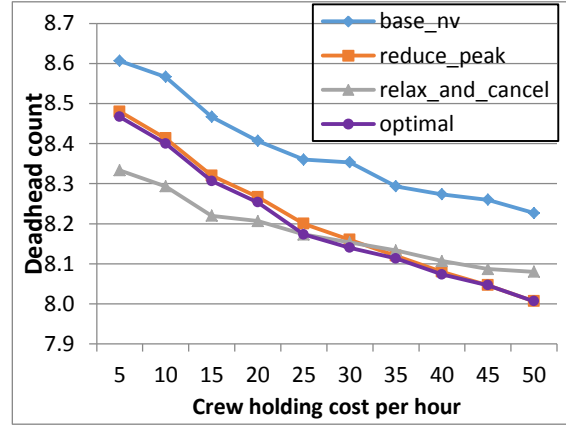
(a) Percentage gap of total cost



(b) Average crew holding time



(c) Average train delay time



(d) Average number of inbound deadheads

Figure 3.9 Performance sensitivity to crew holding cost (IBCP)

Next we keep  $H$  and  $C$  at their default value and vary train delay cost from 100 to 1000 per hour at 100 increments. Train delay cost mainly impacts how expensive it is to delay a train. As shown in Figure 3.10 (b) to (d), all methods generate solutions with more deadheads, lower average train delay time, and higher average crew holding time. Such changes are most obvious when we increase train delay cost from 100 to 200 per hour, and gradually slows down as we further increase  $B$ . The changes in the average

total cost gap are relatively steady in all three heuristics. In particular, `reduce_peak` method maintains high performance in all scenarios of  $B$  values.

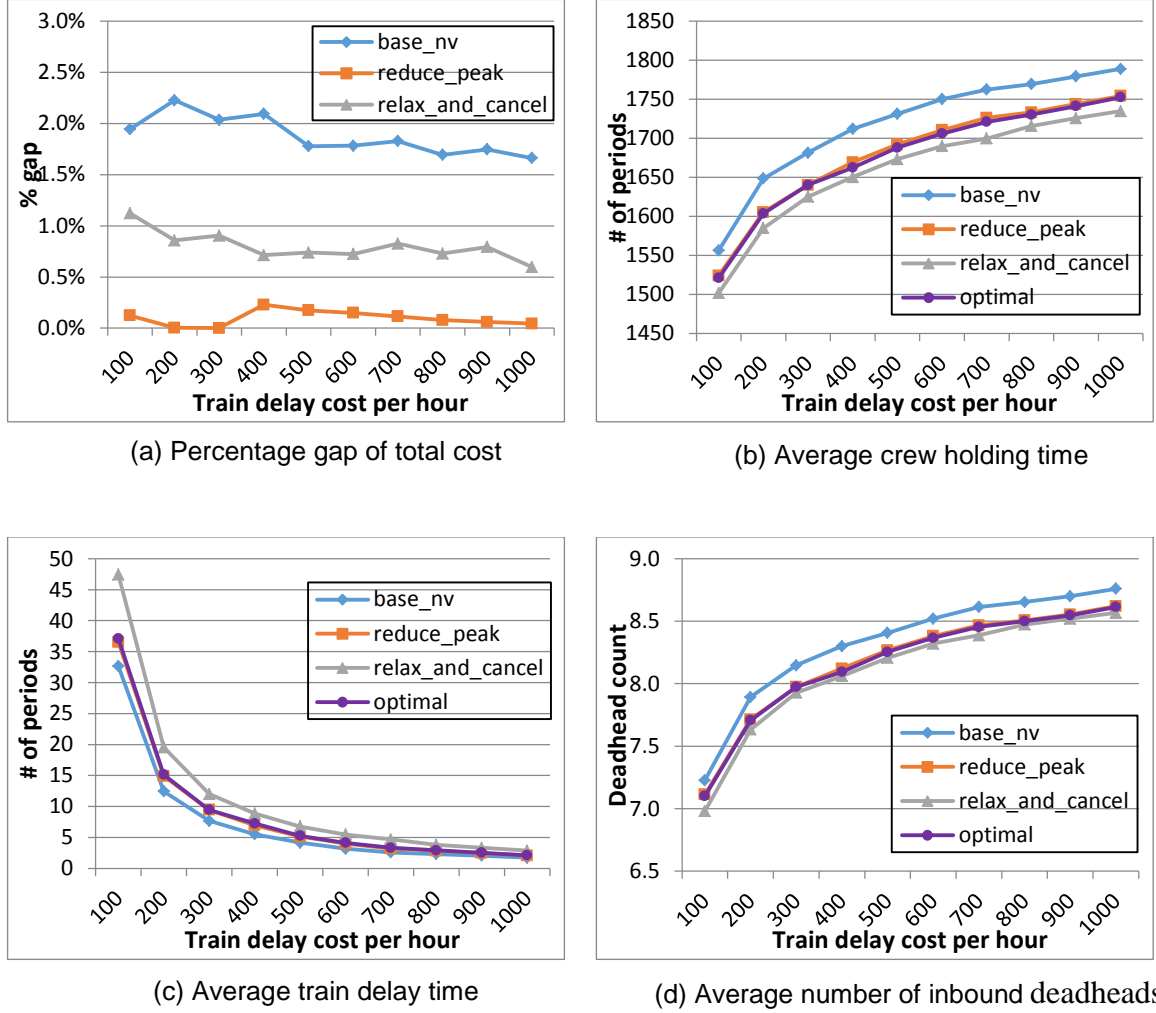


Figure 3.10 Performance sensitivity to train delay cost (IBCP)

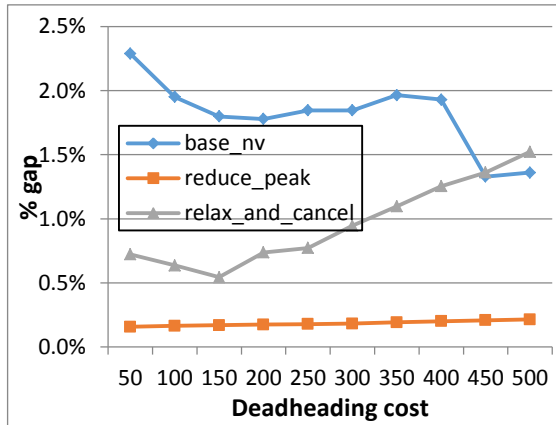
Finally, we study the effect of varying deadheading cost while keeping  $H$  and  $B$  at their default values. For `base_NV`, the value of  $C$  impacts the calculation of last ordering period  $k$  and cumulative ordering quantity  $X_k^*$ . When  $C$  is larger,  $k$  will be

smaller, that is, we will stop ordering additional supplies earlier. In this case, the sequence  $\mathbf{X}^*$  from Step 5 of this algorithm is shorter, and potentially has lower degree of non-monotonicity. In fact, average degree of non-monotonicity decreased from 1.15 to 0.78 as we increase  $C$  from 50 to 500. Naturally, we can expect that the average cost gap of base\_NV will reduce when  $C$  increases as shown in Figure 3.11 (a). For relax\_and\_cancel,  $C$  value does not impact its initial solution because the initial solution only depends on  $\rho$ . However, when  $C$  is large, this heuristic tends to cancel too many deadheads, thus incur more train delays (see Figure 3.11 (c) and (d)). Figure 3.11 (a) shows a clear trend of increasing cost gap as  $C$  increases. Finally, the average cost gap of reduce\_peak only increases slightly with  $C$ . This method continues to find near optimal or optimal solutions for most of the instances regardless of the changes in the parameter values.

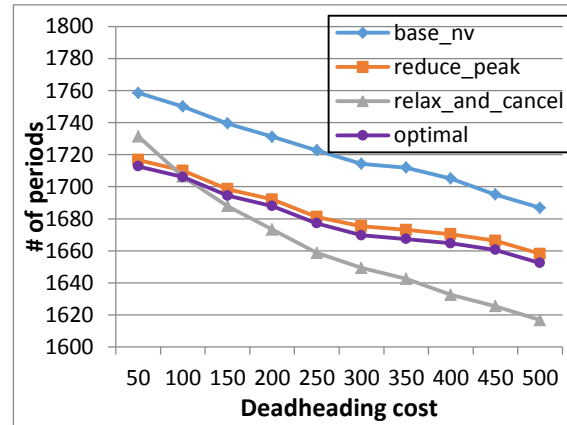
We conclude this section with some key observations obtained from the results of the experiments:

- The performance of base\_NV is closely related to the extent to which net demands violate the assumption of stochastic dominance. This method has good performance when there are few peaks and valleys in the initial sequence of  $\mathbf{X}^*$ ;
- Base\_NV tends to order more inbound deadheads than the optimal solution so that its performance worsens when crew holding is expensive;
- Method reduce\_peak can effectively modify the deadheading plan of base\_NV and significantly improve the solution quality. Moreover, this method is able to deliver promising performance under various settings of cost parameters;

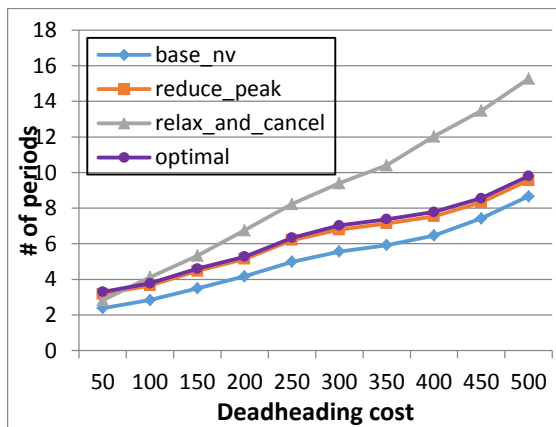
- The performance of relax\_and\_cancel mainly depends on whether the particular setting of cost parameters makes it more likely to cancel wrong pairs of inbound and outbound deadheads. Its cost gap noticeably increases when crew holding cost is very low or deadheading cost is very high.



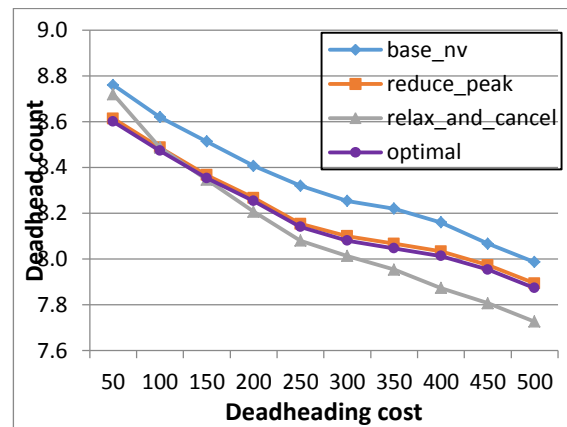
(a) Percentage gap of total cost



(b) Average crew holding time



(c) Average train delay time



(d) Average number of inbound deadheads

Figure 3.11 Performance sensitivity to deadheading cost (IBCP)



### 3.7.6 Compare solution methods for model [IOCP]

For model [IOCP], we consider the recursive method proposed in Section 3.6.1, and the newsvendor-based heuristic `relax_and_cancel` (Section 3.6.2). In addition, we introduce a third method that assumes train arrival/departure times are deterministic. For each instance with stochastic train schedules, we construct a corresponding deterministic instance where each train's actual arrival/departure time is fixed at its expected value, i.e.,  $\text{expected arrival/departure time} = \text{base arrival/departure time} + \text{mean schedule deviation}$ . The deterministic instance can also be solved using the recursive method of Section 3.6.1. The only difference is that the objective function is no longer expected cost but rather a deterministic value for any given plan  $(\mathbf{X}, \mathbf{Y})$ . We call this heuristic *deterministic\_schedule*. As we did in the last section, we will first compare the performance of different testing groups under the default setting of parameters, and then vary some of the parameters to study performance sensitivity.

#### 3.7.6.1 Performance comparison with default parameters

Table 3.3 compares the objective values of model [IOCP] under deadheading plans obtained from different solution methods. The recursive method described in Section 3.6.1 generates the optimal deadheading plan. The cost gaps of the two heuristics, `relax_and_cancel` and `deterministic_schedule`, are calculated as a percentage of the optimal objective value. The minimum gap of `relax_and_cancel` was 0 for all five instance groups, indicating this method was able to find an optimal deadheading plan for some instances in all groups. On the other hand, `deterministic_schedule` failed to find the optimal deadheading plan for all instances of groups B to E. From the perspective

of maximum gap and average gap, the `relax_and_cancel` method also well outperforms `deterministic_schedule`. In particular, `deterministic_schedule` could end up with gaps over 50% of the optimal total cost. The performance of both heuristics is relatively uniform for different instance groups. Although the average gap of group D seems lower than those of other groups, it is mostly caused by the fact that the instances of group D have high sunk delay cost. These instances have high demand but low supply in the first half of the horizon. Due to the lead time of inbound deadheads, the train delay cost in periods before  $l$  is essentially sunk delay cost.

Instance group	Relax_and_cancel			Deterministic_schedule		
	Min	Max	Avg.	Min	Max	Avg.
<b>A</b>	0.0%	12.6%	1.8%	0.0%	33.1%	12.9%
<b>B</b>	0.0%	8.4%	1.1%	0.9%	40.3%	15.3%
<b>C</b>	0.0%	4.7%	0.9%	1.0%	42.7%	18.9%
<b>D</b>	0.0%	1.4%	0.1%	0.6%	43.0%	7.3%
<b>E</b>	0.0%	6.9%	1.2%	0.6%	53.9%	15.3%

Table 3.3 Heuristics for [IOCP]: percentage gap of total cost

Figure 3.12 compares the number of deadheads and average deadheading time of solutions from `deterministic_schedule` and optimal solutions. We can see that the `deterministic_schedule` method tends to have more outbound deadheads (0 to 4 more than the optimal solution) and -1 to 3 more inbound deadheads. In general, the heuristic solution deadheads inbound later and outbound earlier, resulting in a lower average supply level compared to the optimal solution. The gap of supply level ranges from -16.1% to 0.8%, with an average of -5.2%.

As we fix the train schedules at their expected value for deterministic\_schedule method, this heuristic tends to underestimate both crew holding cost and train delay cost. If we plug the deadheading plan of this heuristic into (a) the hypothetical instance with deterministic schedule, and (b) the actual instance with stochastic schedules, and then compute the (expected) total crew holding time and train delay time, we can see that both metrics are, in general, higher when evaluated using the actual instance with stochastic train schedules. Figure 3.13 plots the differences in crew holding time and train delay time (total expected crew holding/train delay time evaluated in the stochastic setting minus that from the deterministic schedule setting). We sort the instances in the order of increasing percentage gap of total cost. The figure shows that the crew holding time was underestimated in all instances while train delay time was also underestimated in many instances. Further, the amount of underestimation seemed to be higher for instances with higher gap in total cost.

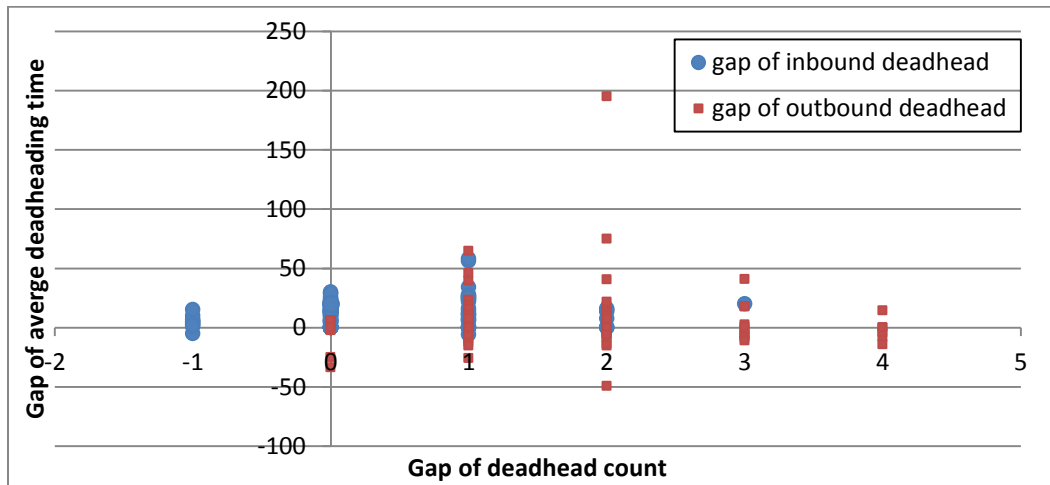


Figure 3.12 Deterministic\_schedule: gaps of deadheading counts and time

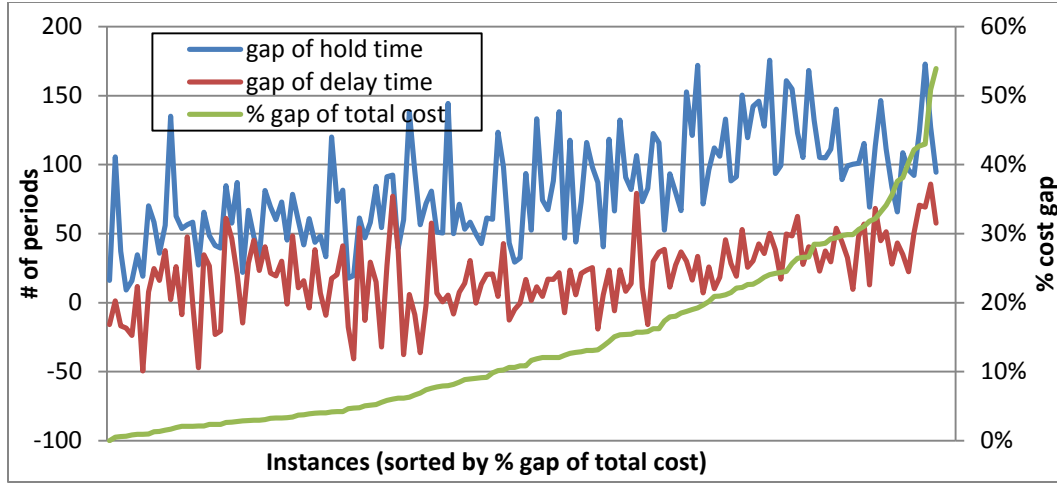


Figure 3.13 Deterministic\_schedule underestimates crew holding and train delay

### 3.7.6.2 Sensitivity to cost parameters and schedule variability

In this section, we examine how the performance of `relax_and_cancel` and `deterministic_schedule` change as we vary the cost parameters  $H$  and  $B$ , and the variance in train schedule. We will compare the percentage gap of total cost with respect to the optimal objective value, total crew holding time, total train delay time, and average crew supply level. In particular, we define average crew supply level as  $\bar{P} := \sum_{t=1}^T P(S_t + X_t \geq D_t + Y_t) / T$ . All metrics were averaged over 150 testing instances.

First, we change crew holding cost from 5 to 50 per hour at increments of 5;  $B$  and  $C$  are at their default values. The results are shown in Figure 3.14. Both heuristics' performance, measured in terms of percentage gap of total cost, was not quite sensitive to crew holding cost because train delay cost is the dominant contributor to the total cost. But Figure 3.14 (c) and (d) reveal that `deterministic_schedule` had noticeably more train delay and lower average crew supply level when  $H$  increases. The optimal

solution and relax\_and\_cancel solution, in contrast, had a much lower increase in train delay time and decrease in crew supply level. Overall, relax\_and\_cancel can adjust its solution quite well as  $H$  changes, but deterministic\_schedule tends to have slightly worse performance with large values of  $H$ .

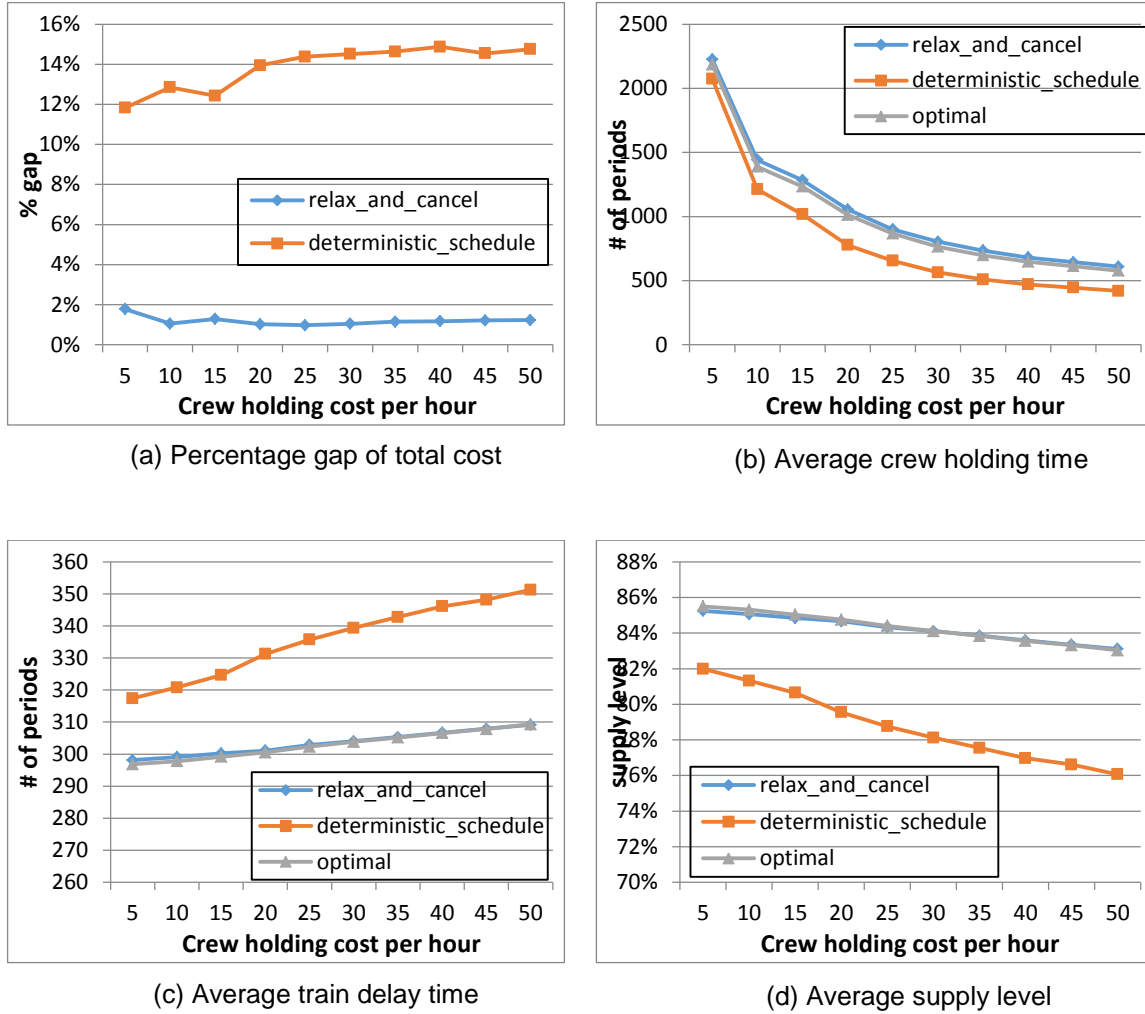


Figure 3.14 Performance sensitivity to crew holding cost (IOCP)

Next, we vary train delay cost in the range of 100 to 1,000 per hour at increments of 100 while keeping  $H$  and  $C$  at their default values. As shown in Figure 3.15 (a), the

total cost gap of method `relax_and_cancel` reduces from 1.7% to 1.1% and is relatively insensitive to train delay cost. However, the gap of `deterministic_schedule` increases nearly linearly. As this method tends to underestimate expected train delay time, it is not surprising that its performance worsens when delaying train is more expensive. In fact, Figure 3.15 (b) to (d) show that `deterministic_schedule` always maintain a lower crew supply level with lower crew holding cost but higher train delay cost. On the other hand, `relax_and_cancel` was able to well control the risk of train delay. Its solutions have very similar expected train delay time and average crew supply level to the optimal solutions. But note that this method has slightly higher crew holding time than the optimal solution (see Figure 3.15 (b)), which is different from what we observed in Figure 3.11 (b) when we discuss performance sensitivity for model [IBCP]. This is because `relax_and_cancel` tends to generate fewer deadheads than the optimal solutions. For [IBCP], the heuristic solution has lower crew holding time because it has fewer inbound deadheads. In [IOCP], this method generates fewer inbound and outbound deadheads; the gap of outbound deadheads is larger than that of inbound deadheads. Hence, the overall crew holding time is larger than the optimal solutions.

Finally, we check the impact of variability in train schedule. To generate instances with different schedule variances, we use the same base schedule as the 150 testing instances we have been using so far. Then we multiple the standard deviation of the gamma distributions by a factor of  $\alpha$  (std. multiplier), where  $\alpha = 0.2, 0.4, \dots, 2.0$ . The modified gamma distributions are used to generate the schedule deviations of each train. Note that the solutions of `deterministic_schedule` do not change because changing

the schedule variance does not impact the deterministic instance whose train schedules are fixed at their expected values. The performance metrics are plotted in Figure 3.16. As expected, the cost gap of `deterministic_schedule` increased very quickly as schedule variance became larger. The expected crew holding time and train delay time also increased while the average crew supply level decreased. On the contrary, `relax_and_cancel` actually worked slightly better (i.e., smaller cost gap) with larger variance in the schedules. This is because the optimal objective value is higher for instances with larger variance. Figure 3.16 (d) shows that `relax_and_cancel` maintained a very similar level of crew supply as the optimal solution while `deterministic_schedule` had much lower supply level. The gap of supply level of `deterministic_schedule` increased as schedule variance became larger.

Based on the experiment results in this section, we can draw the following conclusions:

- `Relax_and_cancel` method has very promising performance that is quite stable under different settings of cost parameters and schedule variance;
- `Deterministic_schedule` method has significantly worse performance than `relax_and_cancel`. By using deterministic train schedules, this heuristic underestimates the potential crew holding time and train delay risk. Its solutions maintain a lower crew supply level than what it needs to have;
- The performance of `deterministic_schedule` is not very sensitive to crew holding cost. However, the cost gap increases almost proportionally when train delay cost or train schedule variance increases.

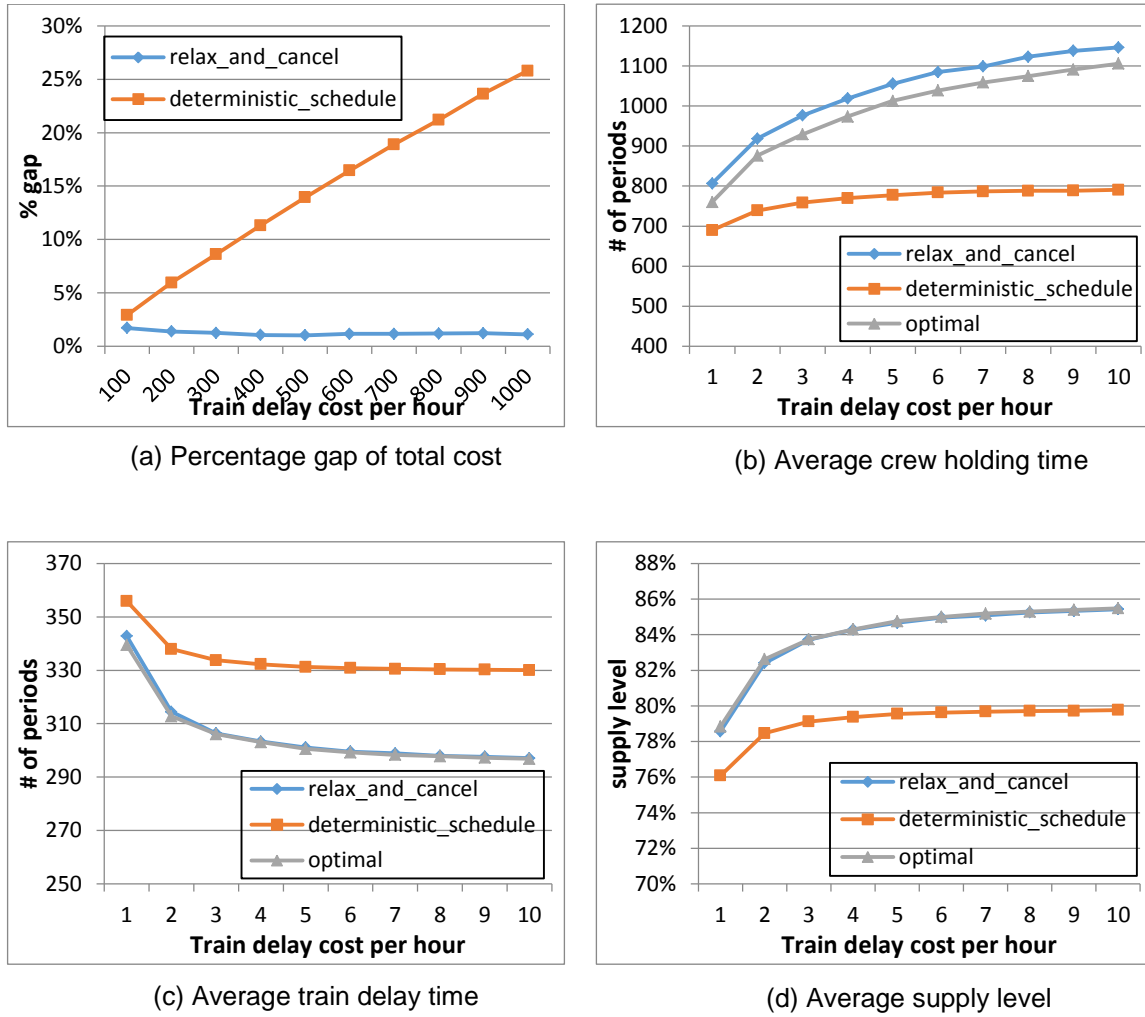


Figure 3.15 Performance sensitivity to train delay cost (IOCP)

### 3.8 CONCLUSIONS

This research studies the problem of deadhead planning with uncertainty in train schedules. We model the problem such that it mimics a multi-period crew inventory planning problem at the away station. Two models were considered: [IBCP] and [IOCP]. The former only considers the option of inbound deadheading while the latter considers both inbound and outbound deadheading. For both models, we propose a

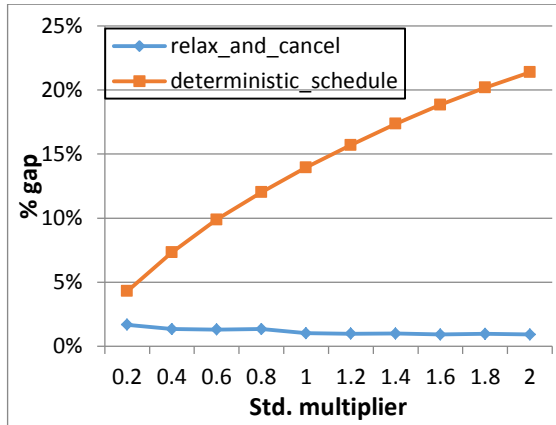


recursive algorithm to solve the problem exactly and several heuristic algorithms that use the similar principle as newsvendor ordering policy (i.e., order up to critical ratio  $\rho$ ). Our computational experiments show that these heuristics are effective and efficient. Moreover, these algorithms are very intuitive: the general idea is to approximately maintain a supply level of  $\rho$  in all periods, and then adjust the deadheading plan (delay or cancel certain deadheads) based on marginal cost analysis. On the other hand, the common practice of using deterministic train schedules may yield solutions that perform poorly when the actual schedules are indeed stochastic, especially when the variance in train schedules are large.

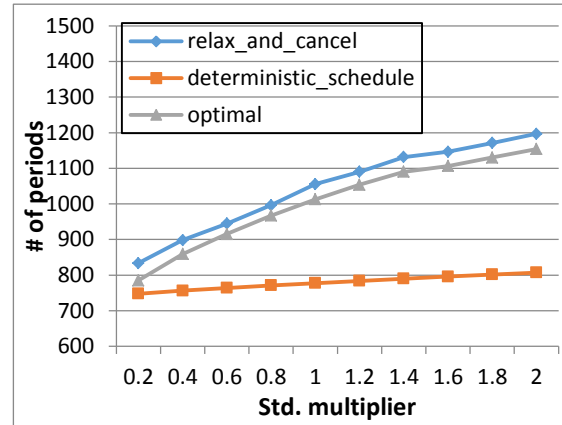
The main contributions of this work include:

s

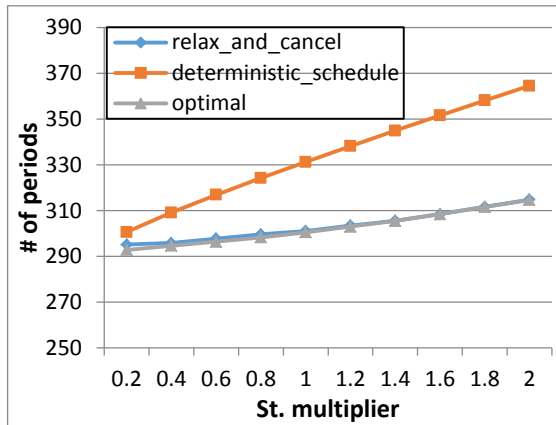
- To our best knowledge, this is the first study in the literature of railroad crew planning that formally models uncertainty in train arrival and departure times;
- We proposed several heuristic algorithms that use the similar ordering principle as the newsvendor problem. These algorithms are intuitive, easy to implement, efficient, and effective;
- The result of Section 3.5.2.1, multi-period inventory problem with continuous demand and unit ordering cost, may be of independent interest (potentially to researchers in the area of inventory control);
- We show that solution methods that assume deterministic train schedules may perform badly in practice when train schedules vary widely.



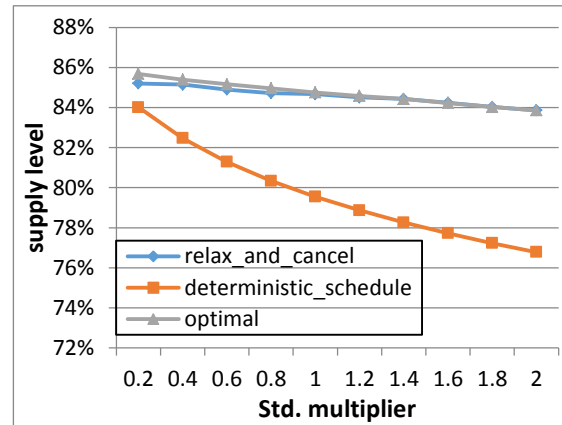
(a) Percentage gap of total cost



(b) Average crew holding time



(c) Average train delay time



(d) Average supply level

Figure 3.16 Performance sensitivity to schedule variance (IOCP)

Future work on this topic may consider relaxing some of our assumptions to make the models more realistic. For example, a more complete treatment of this problem needs to simultaneously consider crew availability at the home and away stations. In practice, we have a fixed-size pool of regular crews at the home station and should only use extra crews sparingly. Therefore, it is important to deadhead crews home not only

for reducing crew holding cost at the away station but also for maintaining crew availability at the home station.

## **Chapter 4: Train Movement Planning**

### **4.1 INTRODUCTION**

In the U.S., over 500,000 carloads of freight and intermodal units, on average, move across the 140,000-mile track network weekly (AAR, 2015(b)). The cars and intermodal units are hauled by trains with different origins, destinations, velocities, and priorities. As trains share the valuable track resource, the goal of movement planning is to coordinate their movements. Effective movement planning is very important to increase track utilization and improve train on-time deliveries of shipments for freight railroads.

For movement planning, the track network is divided into territories that are typically flanked at the two ends by large rail terminals or stations. A territory consists of one or more mainlines connecting the end terminals and a number of intermediate sidings where trains can wait. Trains can travel in either direction on any track, and can move from one track to another at crossover points or junctions at stations and sidings. In a territory with multiple mainlines and siding locations, each train is assigned to one of the parallel tracks. Two trains traveling in opposite directions on a territory can cross each other at any location in the territory only if they are on (traveling or waiting) different tracks. Similarly, one train can pass or overtake another train that is traveling in the same direction only if they are on different tracks. We refer to these two situations as meet and pass events, respectively. For safety purposes, trains must keep a minimum following distance or time gap on each track and at the junctions. An

effective meet-pass plan should not only satisfy all safety requirements, but also consider various operational preferences. For instance, some trains may have specific time window on their arrival and/or departure at certain locations. For a given set of trains scheduled to enter a territory in a planning horizon, movement planning consists of deciding, for each train, the sequence of tracks that the train uses and the timing of its movements so as to meet all safety requirements while minimizing train delays.

Researchers have studied many different versions of the movement planning problem. These studies usually focus on a particular track configuration (single-track vs. multi-track) and traffic pattern (one-way vs. two-way). They may also have different safety requirements and objectives. In this research, we study train movement planning for territories with one or more parallel mainlines. All tracks are bi-directional, allowing trains to travel in either direction. Our objective is to find a set of train *paths*, i.e., sequences of tracks and the corresponding entering and exiting times, that are conflict-free and with minimum overall (weighted) delay.

We discretize the planning horizon into small time intervals, say, one minute periods. Given the layout of a territory consisting of junctions and track segments and the a set of trains projected to enter in the territory, we develop an integer linear program (IP) to model the segment-by-segment movements of each train. To solve this model in a reasonable amount of time, we discuss model strengthening methods and several optimization-based heuristics. We also develop a standalone heuristic that advances trains through the territory by one segment at a time. The goals of this chapter are to: (i) develop an optimization model that can be applied to real-word train dispatching

operations, and (ii) develop heuristic methods that can obtain good solutions within reasonable amount of time.

The rest of this chapter is organized as follows. Section 4.2 provides a review of the research literature on movement planning. We describe our problem in detail and present the mathematical formulation in Sections 4.3 and 4.4, respectively. Sections 4.5 to 4.6 discuss optimization-based and standalone heuristics to solve the problem. We present computational results of all solution methods in Section 4.7 and conclude in Section 4.8.

## **4.2 LITERATURE REVIEW**

Over the last three to four decades, researchers have studied a number of problems that are similar to train movement planning, including train scheduling, train timetabling, and conflict resolution. All of these problems generate conflict-free train movements, constrained by various safety rules. However, these problems model the underlying track network at different levels of detail. For instance, train scheduling and train timetabling focus on the construction of a feasible schedule of train arrival and departure times at stations, hence, may ignore the exact movements within the stations.

Several papers have shown that even simple versions of such scheduling problems are *NP*-complete (e.g., see Caprara et al., 2002 and Lu et al., 2004). Researchers have investigated various models and solution approaches, including continuous-time and discrete-time optimization models, metaheuristics, and simulation. We next provide a review of these studies and categorize them by their modeling approaches. In the following discussion, a “*route*” refers to the sequence of track segments that a train

passes, and a “*path*” refers to the route a train takes and the corresponding entering and exiting times at each segment.

One natural approach to model the movements of trains is to define time variables associated with the entry and exit of trains at each track segment. We refer to this type of models as continuous-time models. Such models typically require a set of auxiliary indicator variables to specify the precedence order among trains on each segment. Carey and Lockwood (1995) study the timetabling problem for passenger trains on a single, one-way corridor. They propose an algorithm that solves a sequence of expanding subproblems. Starting with a subproblem with one train, each iteration adds one more train to the current subproblem. The new subproblem fixes the precedence ordering (but not the entering or exiting times) among trains considered in the previous iteration. Carey (1994a) extends the single-corridor model to a track network with multiple one-way mainlines in each direction, and various complex junctions between the mainlines. Carey (1994b) further extends the model to deal with two-way tracks. Dessouky et al. (2006) propose a branch-and-bound (B&B) algorithm to minimize the total transit time of all trains, assuming that train routes are pre-defined and fixed. The authors propose several propagation rules, applied at each node in the search tree, to derive precedence relationships between trains passing through a track. They then iteratively update the time window of train arrivals and departures and tighten the lower bound of the total transit time. To relax the fixed-route assumption, Mu and Dessouky (2011) discuss two heuristic algorithms. Their first heuristic preselects a set of operationally preferred routes and solves an IP model that allows trains to select only

among these routes. The second method iteratively generates routes using a genetic algorithm, and solves a fixed-route problem.

Another stream of research adopts modeling concepts and solution approaches from job-shop scheduling problems. Trains are treated as jobs that require a number of operations, where an operation represents the movement of a train through a track section. Each track section is analogous to a zero-buffer machine that allows one train to pass at a time. The processing time of an operation is the minimum traversal time required or the scheduled dwelling time on a track section. The objective function minimizes the total time needed to “process” all trains on the specified track network (i.e., the makespan). D’Ariano et al. (2007) consider the problem of finding a new feasible timetable when the original one is disrupted, assuming that trains still run on their original routes. Based on the alternative graph representation, proposed by Mascis and Pacciarelli (2002), they develop a B&B algorithm that gradually expands a partial selection of alternative arcs. Each branching creates two child nodes by choosing a pair of unselected alternative arcs and adding one of them to the selection of the parent node. Each child node further augments its selection by adding more alternative arcs based on the current selection. D’Ariano et al. (2008a) show that a flexible initial timetable that only specifies time windows, but not the exact times, of arrivals and departures at stations can produce better solutions when disruptions occur. D’Ariano et al. (2008b) present an iterative two-stage approach to relax the fixed-route assumption during rescheduling. Stage 1 optimizes the fixed-path timetabling problem, and Stage 2 searches for promising alternative routes using a local search algorithm. Liu and Kozan (2009) formulate train



scheduling as a parallel machine job-shop scheduling problem with blocking constraints. They propose an iterative procedure that adds one more train to the subproblem each time. To solve the subproblem, they first drop the blocking constraints and apply the shifting bottleneck algorithm. The solution is feasible if it does not violate any blocking constraint; otherwise, they recover the solution of the previous subproblem and insert the movements of the newly added train. Liu and Kozan (2011) further introduce the no-wait constraint to model the non-stopping requirement on prioritized trains. Their heuristic algorithm constructs the path of one train at a time, and uses the best-insertion-heuristic to find the best order of trains on the bottleneck track section.

We next discuss discrete-time models that are similar to our model. Brännlund et al. (1998) consider the problem of selecting profitable train service requests and scheduling the selected trains along a single mainline track. Their Lagrangian relaxation method dualizes the track capacity constraints. The relaxed problem is decomposed by train, and each subproblem reduces to a shortest path problem in a time-space network with Lagrangian arc cost. Caprara et al. (2002) consider a one-way corridor, where stations are aggregated into uncapacitated nodes. Their time-space network creates arcs between each arrival and departure node pair at a station, and between each departure and arrival node pair at two adjacent stations. Their model imposes non-overtaking requirements on track segments between adjacent stations and minimum following distance requirements for consecutive arrivals/departures at stations. Caprara et al. (2006) discuss the modeling of several real-world operation issues, such as finite station capacity and track maintenance. With the assumption that the run time of

each train on each segment is known and fixed, they can reduce the size of the time-space network, and strengthen the non-overtaking constraints proposed in Caprara et al. (2002). Cacchiani et al. (2010) extend the models of Caprara et al. (2002 and 2006), and address train dispatching in a track network with multiple one-way or two-way mainlines between stations. Cacchiani et al. (2008) propose a path-based formulation and use binary variables to represent the selection of paths in the time-space network. They propose an iterative procedure that applies column generation and dynamic constraint generation to solve the LP relaxation. They then generate integral solutions based on the LP solution using a heuristic algorithm and several local search methods. Şahin et al. (2008) propose a multi-commodity network flow model. Their time-space network models stations as capacitated nodes, and constructs travel arcs between adjacent stations and waiting arcs within each station. Their IP-based heuristic iteratively reduces the maximum delay allowed for each train and solves the restricted IP model. They also discuss a heuristic that sequentially resolves conflicts in the unimpeded train schedules. For each conflict, they solve two LP subproblems to determine the train that should yield to another train. This approach generates better solutions than cost-based greedy approaches, but is very expensive in computational time. Harrod (2011) uses hyper-arcs to represent the movements of trains. This model explicitly considers the transition of trains between adjacent track sections in order to avoid unrealistic movements of two trains traveling in opposite directions switching their locations simultaneously. Balakrishnan et al. (2012) discuss train dispatching for single-track territories in U.S. freight railways. They propose several types of valid inequalities that can effectively

strengthen their model formulation and reduce the computational time of the B&B procedure. Our work adopts similar modeling framework as Balakrishnan et al. (2012), but specifically models the movements from one track segment to the next one to capture train headway requirement at junctions.

Besides the optimization-based models discussed above, several studies use metaheuristics (e.g., Corman et al., 2010) and discrete-event simulation (e.g., Lu et al., 2004). Cordeau et al. (1998) present a broad survey of train scheduling models, and categorize them as fixed-velocity models and variable-velocity models. Lusby et al. (2009) provide a more recent and extensive review of studies for different track configurations.

### 4.3 PROBLEM DESCRIPTION

Movement planning entails coordinating the movements of trains travelling through a territory and provides the detailed routes and timetables that each train should follow. A territory (typically) consists of one or more mainline tracks connecting two terminals, several sidings along the mainlines, and various junctions linking the mainlines and sidings. We call the junctions *control points* and the tracks linking two adjacent control points *segments*. A segment is a portion of single track; parallel tracks connecting the same two control points are called parallel segments. Each segment may consist of one or more *sections* that are controlled by traffic signals. Each section can accommodate only one train at a time, a requirement that is imposed in order to ensure adequate separation between consecutive trains. Figure 4.1 illustrates the representation of a track network using control points and segments. Nodes *A*, *H*, *I*, and *R* are dummy

control points representing the end points of the territory. The combined node ( $B, C, J, K$ ) is a double crossover that allows trains to switch from one mainline to the other from either direction. Control points ( $D, E, L, M, N$ ) and ( $F, G, O, Q, P$ ), respectively, represent a single crossover between the two mainlines and a switch between the lower mainline and the siding. The arcs connecting control points are called segments.

For a given planning horizon, we have a set of trains that are either initially in the territory or anticipated to enter the territory at a future time. A train profile specifies a train's starting and ending segments in the territory, its release or available entry time at the starting segment, velocity, and priority. Some trains may have time window restrictions on their arrival/departure at certain locations. For instance, passenger trains should not leave a passenger pickup location before its scheduled departure time. Since trains share track resources in the territory, we need to coordinate their movements so that they do not conflict with each other. First, we assume that trains can only travel in one direction from its origin to destination and cannot back up. Therefore, we should avoid deadlocks where trains in opposite directions appear on the same segment. Second, we need to maintain a minimum following distance between trains traveling in the same direction. The traffic signaling system automatically maintains this following distance since each section can only hold one train at a time. Finally, trains passing through a control point need to be separated by a minimum headway time if they share any track segments. This headway time allows a train to be clear of the control point after it enters the control point. In Figure 4.1, trains traveling through control point arcs  $DE$  and  $DM$  should be separated by headway time since these movements share segment

$CD$ . However, two trains using arcs  $DE$  and  $LM$ , respectively, can travel simultaneously. We call a set of segment pairs a *headway set* if trains moving across these segment pairs need to be separated by headway time. For instance,  $\{(CD, MO), (MO, CD), (CD, EF), (EF, CD)\}$  is a headway set.

In order to avoid deadlocks and maintain safety requirements, we may need to detour a train from one mainline to another mainline or a siding. Trains may have different traversal time on parallel segments because of the different track speed restriction. There is also additional traveling time incurred for using the control point arcs as trains need to slow down while using crossovers and switches. Trains may also need to stop and wait on a segment for a certain amount time before proceeding to the next segment. For simplicity, we assume that trains travel at a constant velocity on each segment and can only stop to wait on the last section of a segment. In Figure 4.1, the triangle areas 1 to 9 indicate potential waiting locations for trains traveling from left to right (eastbound). For future trains, we may also hold them at a terminal outside the territory if the territory is too congested to handle more trains. We refer to such delays *source delays*.

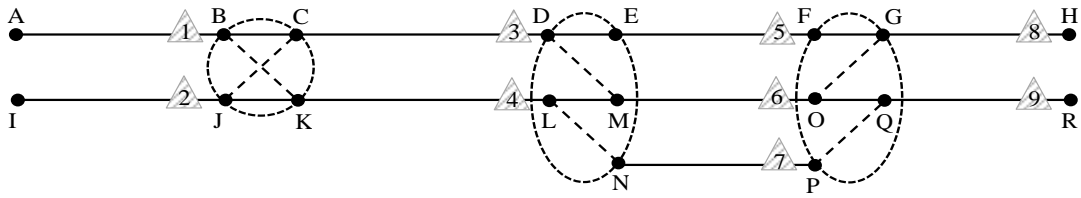


Figure 4.1 Track layout

The objective of movement planning is to generate conflict-free train movements

while minimizing train delays, including source delays, waiting time in the territory, and extra travel time incurred by detouring. In the next section, we propose a discrete-time integer programming model that minimizes total train delays weighted by train priorities.

#### 4.4 MODEL FORMULATION

To describe the track network, let  $P$  be the set of control points and  $S$  be the set of segments.  $Q = QI \cup QF$  is the set of trains that travel in the territory in a given planning horizon, where  $QI$  and  $QF$  are sets of initial trains and future trains, respectively. We discretize time into short (say, 1-minute) periods, indexed from 1 to  $|T|$ . For each train  $q$ , we can calculate its traveling time on each segment  $s$  based on train speed, track speed restriction, and the length of the segment. The set of periods,  $T(s, q)$ , that train  $q$  can enter segment  $s$  can be obtained based on the train's release time at the starting segment and the shortest traveling time from the starting segment to segment  $s$ . We can then adjust  $T(s, q)$  using any time window requirement of train  $q$  at segment  $s$ . Note that  $T(s, q)$  includes period  $|T| + 1$  if  $s$  is an immediate downstream segment to the starting segment of a future train  $q$ ; this option allows the train to be delayed beyond the planning horizon. Since we assume that a train can only wait at the end of each segment, we can also generate the set of periods when train can wait on segment  $s$ ,  $W(s, q)$ , based on  $T(s, q)$ . This end-of-segment-waiting assumption, along with the assumption that trains travel at a constant velocity on segments, enables us to anticipate the location of a train on a segment once we know the time it enters the segment. In other words, if we know that train  $q$  enters segment  $s$  at time  $t$ , then we can easily compute the set of periods that train  $q$  will travel in each section of this segment. To describe the formulation, we

define the following notation:

**Notation:**

$T$ : set of time periods of the planning horizon

$P$ : set of control points

$S$ : set of segments

$h_p$ : number of headway sets at control point  $p$

$H_i(p)$ : the  $i^{\text{th}}$  headway set at control point  $p$ ,  $i = 1, \dots, h_p$

$N(s)$ : set of sections consisting of segment  $s$

$S(q)$ : set of segments that train  $q$  may travel on

$F(s, q)$ ,  $B(s, q)$ : sets of forward (immediate downstream) and backward (immediate upstream) segments of segment  $s$  in the direction of train  $q$

$T(s, q)$ : set of periods when train  $q$  can enter segment  $s$

$W(s, q)$ : set of periods when train  $q$  can wait on segment  $s$

$E(n, q, t)$ : set of periods when train  $q$  can enter a segment so as to travel on section  $n$  at time  $t$

$D$ : set of train directions;  $D = \{\text{eastbound, westbound}\}$

$Q(s)$ : set of trains that may travel on segment  $s$

$Q_d$ : set of trains traveling in direction  $d$

$o_q, e_q$ : starting segment and ending segment of train  $q$

$r_q$ : release time of train  $q$  at the end of its starting segment

$\rho_q$ : priority of train  $q$

$\tau_{sq}$ : minimum time (periods) of train  $q$  to travel through segment  $s$

$\delta_{nq}$ : waiting section indicator;  $\delta_{nq} = 1$  if train  $q$  can wait on section  $n$  (i.e.,  $n$  is the last section of a segment in the direction of  $q$ )

$\mu$ : minimum headway time required at all control points

To model the movements of trains, we define two types of binary variables:

segment entrance variable  $x$ , and waiting variable  $w$ . Specifically, let

$x_{s's}^{qt} = 1$  if train  $q$  enters segment  $s$  in period  $t$  from a previous segment  $s'$ , for all  $q \in Q$ ,  $s$

$\in S(q)$ ,  $s' \in B(s, q)$ , and  $t \in T(s, q)$ ; and,

$w_s^{qt} = 1$  if train  $q$  waits at the last section of segment  $s$  during  $[t, t+1)$ , for all  $q \in Q$ ,  $s \in$

$S(q)$ , and  $t \in W(s, q)$ .

Note that the  $\mathbf{x}$  variables record the last segment  $s'$  that train  $q$  uses before it enters segment  $s$  so as to determine which headway set this movement belongs to. For notational brevity, we define  $\mathbf{x}$  and  $\mathbf{w}$  variables for all  $t \in T$ , and let  $x_{s's}^{qt} = 0$  if  $t \notin T(s, q)$ ,  $w_s^{qt} = 0$  if  $t \notin W(s, q)$ .

The cost coefficient of variable  $x_{s's}^{qt}$  is  $\alpha_{s's}^{qt}$ , which includes (i) penalty of source delay if  $q$  is a future train and  $s' = o_q$ , (ii) detouring cost incurred by traveling from segments  $s'$  to  $s$ , and (iii) penalty for extra traveling time needed on segment  $s$  if train needs to travel at a lower velocity on that segment. The waiting cost is  $\beta_s^{qt}$ , which may depend on the waiting location, start waiting time, and train priority.

Finally, we use a binary indicator variable,  $v_{st}^d$ , to indicate the direction of train flow on segments. The indicator  $v_{st}^d = 1$  if segment  $s$  is assigned to trains traveling in direction  $d$  at time  $t$ , and 0 otherwise, for all  $s \in S$  and  $t \in T$ .

#### 4.4.1 Movement Planning Formulation

We now present the IP formulation using the above decision variables and notation. This formulation extends the single-track model proposed by Balakrishnan et al. (2012).

$$[\mathbf{MP}] \text{ Minimize } \sum_{q \in Q} \sum_{s \in S(q)} \left( \sum_{s' \in B(s, q)} \sum_{t \in T(s, q)} \alpha_{s's}^{qt} x_{s's}^{qt} + \sum_{t \in W(s, q)} \beta_s^{qt} w_s^{qt} \right) \quad (4.1)$$

subject to:



Departure at the starting segment:

$$\sum_{t \in T(s,q)} \sum_{s \in F(o_q,q)} x_{o_q,s}^{qt} = 1 \quad \forall q \in QF, \quad (4.2)$$

$$\sum_{s \in F(o_q,q)} x_{o_q,s}^{q,r_q} + w_{o_q}^{q,r_q} = 1 \quad \forall q \in QI, \quad (4.3)$$

Flow conservation on segments:

$$w_{o_q}^{q,t-1} = \sum_{s \in F(o_q,q)} x_{o_q,s}^{qt} + w_{o_q}^{qt} \quad \forall q \in QI, t \in (r_q, |T|], \quad (4.4)$$

$$\sum_{s' \in B(s,q)} x_{s's}^{q(t-\tau_{sq})} + w_s^{q,t-1} = \sum_{s' \in F(s,q)} x_{ss'}^{qt} + w_s^{qt} \quad \forall q \in Q, s \in S(q) \setminus \{o_q, e_q\}, t \in T, \quad (4.5)$$

Section capacity:

$$\sum_{q \in Q_d \cap Q(s)} \left( \sum_{t' \in E(n,q,t)} \sum_{s' \in B(s,q)} x_{s's}^{qt'} + \delta_{nq} w_s^{qt} \right) \leq v_{st}^d \quad \forall s \in S, n \in N(s), t \in T, d \in D, \quad (4.6)$$

$$\sum_{d \in D} v_{st}^d \leq 1 \quad \forall s \in S, t \in T, \quad (4.7)$$

Control point headway:

$$\sum_{(s',s) \in H_i(p)} \sum_{q \in Q(s): s' \in B(s,q)} \sum_{t'=t}^{t+\mu-1} x_{s's}^{qt'} \leq 1 \quad \forall p \in P, i=1, \dots, h_p, t \in T, \quad (4.8)$$

Integrality:

$$x_{s's}^{qt} \in \{0,1\} \quad \forall q \in Q, s \in S(q), s' \in B(s,q), t \in T, \quad (4.9)$$

$$w_s^{qt} \in \{0,1\} \quad \forall q \in Q, s \in S(q), t \in T, \quad (4.10)$$

$$v_{st}^d \in \{0,1\} \quad \forall s \in S, d \in D, t \in T. \quad (4.11)$$

The objective function (4.1) minimizes the total weighted delay of all trains, including delay at source, detouring cost, and waiting time in the territory. Departure constraints (4.2) ensure that each future train leaves its starting segment on or after its release time, while constraints (4.3) enforce each initial train to either leave its starting

segment at the release time or stop and wait at the starting segment. Constraints (4.4) and (4.5) ensure the continuity of train movements on all segments between its starting segment (inclusive) and ending segment (exclusive) respectively. Specifically, if train  $q$  is at the end of segment  $s$  at time  $t$ , i.e., train  $q$  is ready to either enter a downstream segment at  $t$  or wait on  $s$  from  $t$  to  $(t+1)$ , then it must either enter segment  $s$  ( $s \neq o_q$ ) at time  $(t - \tau_{sq})$  or wait on  $s$  from  $(t - 1)$  to  $t$ . Constraints (4.6) force the flow indicator variable  $v_{st}^d = 1$  if any train in direction  $d$  is traveling on a section of segment  $s$  or waiting on segment  $s$  at time  $t$ . Further, these constraints ensure that at most one train (in direction  $d$ ) can appear on any section of  $s$  at any time  $t$  since the  $v$  variables are binary. Constraints (4.7) then forbid trains in opposite directions to appear on the same segment simultaneously. Constraints (4.8) impose the headway requirement at each control point, that is, no more than one train can pass through a pair of segments in a headway set in any  $\mu$  periods of time. The remaining constraints (4.9) to (4.11) specify that all variables are binary.

#### 4.4.2 Train-based forcing constraints

Balakrishnan et al. (2012) propose several types of valid inequalities to enhance their base formulation. In particular, they show that their *train-based non-concurrency inequality* is very effective for improving LP relaxation bounds. We illustrate this inequality in the time-space diagram of Figure 4.2, where  $x$ -axis refers to the track space and  $y$ -axis shows the timeline. The strings in the diagrams represent the movements of trains: a tilted line represents the movement on a segment ( $x$  variable) and a vertical

arrow represents waiting at the end of a segment ( $w$  variable). If a train  $q$  enters segment  $s$  during  $M(s, q, t) = [t - \tau_{sq} - \mu + 1, t]$ , then this train either occupies segment  $s$  at time  $t$  or exits from it after period  $t - \mu$ . Therefore, segment  $s$  is not available to trains traveling in the opposite direction of  $q$  at time  $t$ . The authors propose a set of forcing constraints that enforce  $v_{st}^d$  to be one if any train in direction  $d$  enters segments  $s$  during  $M(s, q, t)$ .

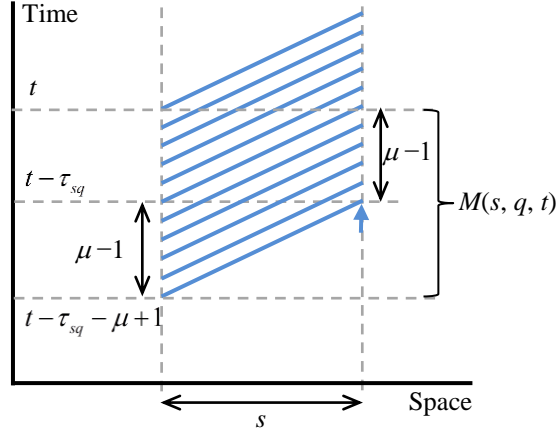


Figure 4.2 Illustration of train-based forcing constraints

These forcing constraints are also valid in our model. But we can strengthen them by incorporating waiting variables. Note that Balakrishnan et al. (2012) does not consider waiting on a single track segment and thus this enhancement does not apply. If train  $q$  waits on segment  $s$  at  $t - \mu$ , it must have entered this segment before the first period in  $M(s, q, t)$  and will exit after  $t - \mu$ . Therefore,  $w_s^{q(t-\mu)}$  is not indicated by any  $x$  variables in the set  $M(s, q, t)$  and it prevents trains in the opposite direction from using

segment  $s$  at  $t$ . The strengthened inequality with waiting variable is shown in (4.12).

$$\sum_{t' \in M(s, q, t)} \sum_{s' \in B(s, q)} x_{s's}^{qt'} + w_s^{q(t-\mu)} \leq v_{st}^d, \quad \forall s \in S, t \in T, d \in D, q \in Q_d. \quad (4.12)$$

#### 4.4.3 Headway-based forcing constraints

Headway-based forcing constraints are valid inequalities that link the segment-level headway requirement with the segment flow indicators. As illustrated in Figure 4.3, at most one train can enter or exit from a given segment during  $[t - \mu + 1, t]$ . Further, if any train  $q$  enters or exits the segment during  $[t - \mu + 1, t]$ , it makes the segment unavailable to trains in the opposite direction at time  $t$ , i.e.,  $v_{st}^d = 1$  and  $v_{st}^{d'} = 0$ , where  $d$  is the direction of train  $q$  and  $d'$  is the opposite direction of  $d$ . The following constraints (4.13) and (4.14) are headway-based forcing constraints at the entry and exit of a segment, respectively. Note that although these two sets of constraints have similar structure, they are not equivalent. We will compare their impact on the model strength and computational time in Section 4.7. Balakrishnan et al. (2012) also discuss headway-based valid inequality that has similar form as constraint (4.13). They do not have constraint (4.14) because their movement variables are defined differently and do not capture the previous segment.

$$\sum_{q \in Q_d(s)} \sum_{s' \in B(s, q)} \sum_{t' = t - \mu + 1}^t x_{s's}^{qt'} \leq v_{st}^d, \quad \forall s \in S, t \in T, d \in D, \quad (4.13)$$

$$\sum_{q \in Q_d(s)} \sum_{s' \in F(s, q)} \sum_{t' = t - \mu + 1}^t x_{ss'}^{qt'} \leq v_{st}^d, \quad \forall s \in S, t \in T, d \in D. \quad (4.14)$$

The headway-based forcing constraints, together with the train-based forcing

constraints, are useful to eliminate some fractional solutions where multiple trains' paths split into two strings on parallel segments (partial detour). Hereafter, we refer to formulation (4.1) – (4.12) the *base model*. The headway-based forcing constraints are used in the rounding heuristic discussed in the next section.

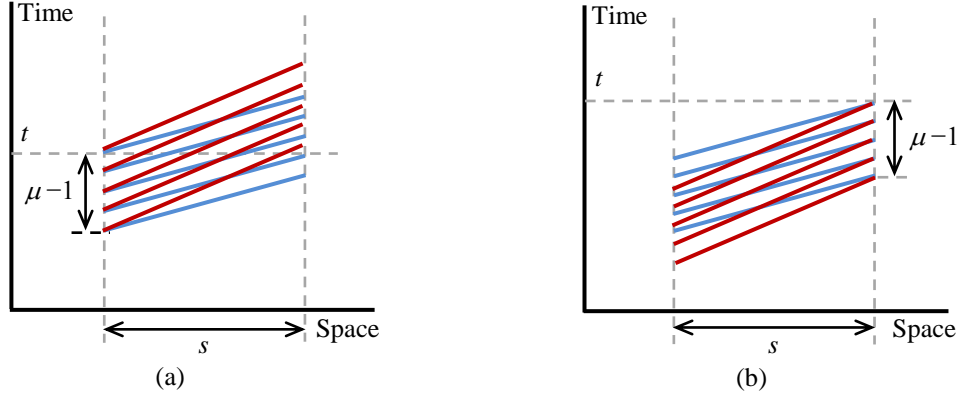


Figure 4.3 Illustration of headway-based forcing constraints

(a) at most one train enters  $s$  in any  $\mu$ -period interval (b) at most one train exits from  $s$  in any  $\mu$ -period interval

Model [MP] is a large-scale optimization problem and is generally difficult to solve for real-life problem instances. In the next two sections, we propose several optimization-based heuristics and a greedy standalone heuristic.

#### 4.5 OPTIMIZATION-BASED HEURISTICS

In this section, we propose five heuristic algorithms for the optimization model [MP]. The first method reduces the model size by narrowing the time window during which a train can enter each segment. The second and third methods start with solving the LP relaxation of model [MP], and iteratively impose integrality requirements on a subset of variables or fix their values to integers (0 or 1). The last two methods explore

the idea of “divide-and-conquer”. We iteratively solve subproblems that include either a subset of trains or a shorter planning horizon.

#### **4.5.1 LP-based time window method**

In this section, we propose a heuristic to restrict the time window during which a train can enter or wait on a segment so as to reduce the model size. In model [MP], we allow a train to enter a segment any time after its release time from the starting segment plus the minimum time it needs to travel from the starting segment to the given segment. Therefore, the number of variables and constraints increase linearly with the horizon length. If we know the maximum delay a train possibly needs in the optimal solution, we can reduce the sizes of  $T(s, q)$  and  $W(s, q)$  and thus reduce the model size. We propose a heuristic to estimate each train’s maximum delay based on an LP solution with fractional train paths.

The time window heuristic first solves the LP relaxation of the base model. It then fixes the paths that are integral and imposes a time window for the trains with fractional paths. The restricted problem is then solved as an IP. For a given LP solution, a train may have multiple fractional paths that use different parallel tracks and/or have different amount of delays due to conflicts with other trains. At a minimum, our time window will allow the train to take its most delayed fractional path. But such time windows are usually too tight and are not sufficient to obtain an integral solution. We next propose a method to extend the minimum time window by considering different overtaking scenarios among trailing train pairs.

For a pair of trains traveling in the same direction, the LP relaxation tends to take

advantage of partial overtaking by sending both trains on fractional paths. However, in an integral solution, either the following train overtakes the leading train using parallel track resources or they maintain the order. Here we only consider overtaking scenarios at the nearest sidings before (upstream) and after (downstream) the location where partial overtaking first happens. Figure 4.4 illustrates the fractional paths of eastbound trains,  $q_1$  and  $q_2$ . The two trains' fractional paths start to overlap at control point  $p$ . Train  $q_1$  is released earlier but has is slower than  $q_2$  (see Figure 4.4 (a)). In one scenario, we can let  $q_1$  travel unimpeded and let  $q_2$  follow  $q_1$  all the way till its ending segment. In this case,  $q_2$  will be delayed before passing control point  $p$  for the amount of time that will ensure it does not catch up  $q_1$ , shown as  $delay(q_2, q_1)$  in Figure 4.4 (b). Figure 4.4 (c) shows another scenario where we do not delay  $q_2$  but let  $q_1$  yield to  $q_2$  at the previous siding location  $s_1$ . The amount of delay time  $delay(q_1, q_2)$  will ensure the headway between  $q_1$  and  $q_2$ .

A more sophisticated approach is to consider different overtaking scenarios based on the relative priorities of the two trains involved in partial overtaking. If the leading train  $q_1$  has lower priority, it will most likely be overtaken at the upstream siding  $s_1$  as shown in Figure 4.4 (b). But if  $q_1$  has higher priority, we consider the option of overtaking at the downstream siding  $s_2$  so that  $delay(q_1, q_2)$  will be smaller at the expense of delaying  $q_2$ . To compute delay time of the following train  $q_2$ , we consider the scenario of it following all the way to the end if it has lower priority, as shown in Figure 4.4 (c). If  $q_2$  has higher priority, it may follow only up to the downstream siding  $s_2$  and then overtake  $q_1$ . The delay time  $delay(q_2, q_1)$  is smaller in the latter case. We present

the pseudo code to compute  $delay(q_1, q_2)$  and  $delay(q_2, q_1)$  in the procedure **compute\_time\_window**. The final time window of train  $q_1$  is the maximum value among all  $delay(q_1, q_2)$  where  $(q_1, q_2)$  is a pair of trains of same direction whose fractional paths overlap in the LP solution.

**Compute\_time\_window:**

Step 1: Solve the LP relaxation of the base model

For all  $q \in Q$  and  $s \in S(q)$ , let:

$t_0(s, q) :=$  the earliest time train  $q$  can enter segment  $s$  (based on train release time and shortest traveling time from  $o_q$  to  $s$ )

$t_m(s, q) :=$  the earliest time that train  $q$  enters segment  $s$  (partially) in the LP solution

Step 2: Get the maximum delay of each train in the LP solution

For each train  $q$

If train  $q$  has a unique and integral path in the LP solution

Let  $var(q) :=$  set of  $x$  and  $w$  variables consisting of the solution path of  $q$

Else

Let  $delay(q) :=$  amount of delay incurred on  $q$ 's most delayed fraction path

End If

End

Step 3: Estimate delay needed for solving passing conflicts

Let  $QO :=$  set of train pairs  $(q_1, q_2)$  that travel in the same direction and whose fractional paths overlap; assume  $q_1$  is initially ahead of  $q_2$  (i.e.,  $q_1$  is leading)

For each train pair  $(q_1, q_2) \in QO$ , let

$s :=$  the first segment where the fractional paths of  $q_1$  and  $q_2$  overlap

$s_1 :=$  the last siding before  $s$  (upstream siding)

$s_2 :=$  the first siding after  $s$  (downstream siding)



If  $\rho_{q_1} > \rho_{q_2}$

Estimate delay time of  $q_1$  if it yields to  $q_2$  at  $s_2$  ( $q_2$  overtakes  $q_1$  at  $s_2$ ):

$$estimate := tm(s_2, q_1) + \tau_{s_2, q_2} - \tau_{s_2, q_1} + 2\mu$$

$$delay(q_1) := \max\{delay(q_1), estimate - t_0(s_2, q_1)\}$$

Estimate delay time of  $q_2$  if it follows  $q_1$  until the ending segment or the last commonly used segment (no overtaking):

Let  $s'$  = the last segment that  $q_1$  and  $q_2$  may both take

$$delay(q_2) := \max\{delay(q_2), t_m(s', q_1) + \mu - t_0(s', q_2)\}$$

Else

Estimate delay time of  $q_1$  if it yields to  $q_2$  at  $s_1$  ( $q_2$  overtakes  $q_1$  at  $s_1$ ):

$$estimate := t_m(s_1, q_2) + \tau_{s_1, q_2} - \tau_{s_1, q_1} + \mu$$

$$delay(q_1) := \max\{delay(q_1), estimate - t_0(s_1, q_1)\}$$

Estimate delay time of  $q_2$  if it follows  $q_1$  up to  $s_2$  ( $q_2$  overtakes  $q_1$  at  $s_2$ ):

$$delay(q_2) := \max\{delay(q_2), t_m(s_2, q_1) + \mu - t_0(s_2, q_2)\}$$

End If

End

Step 4: Update time windows

For  $q \in Q$  and  $s \in S(q)$ , let  $T_{res}(s, q)$  and  $W_{res}(s, q)$  be the restricted time windows:

$$T_{res}(s, q) := [t_0(s, q), t_0(s, q) + delay(q)]$$

$$W_{res}(s, q) := [t_0(s, q) + \tau_{sq}, t_0(s, q) + delay(q) + \tau_{sq} - 1]$$

End

After obtaining the restricted time window of each train as described in the procedure *compute\_time\_window*, we then solve the restricted model [MP] as an IP. Note that the restricted model is not guaranteed to be feasible. To avoid infeasibility, we allow a train to be “cancelled” or delayed beyond the planning horizon if it cannot find a feasible (integral) path. If a train is cancelled because of its tight time window, we will relax a subset of trains’ time windows by  $\delta$  periods and resolve. This subset of

trains includes the train that was cancelled and other trains in the same direction whose fractional paths overlap with the cancelled train in the LP solution. This iterative time window relaxation procedure is describe in the procedure *time\_window\_heuristic*.

***Time\_window\_heuristic:***

Step 1: Build the restricted [MP] model (IP)

For each train  $q$

If train  $q$  has a unique and integral path in the LP solution

Create  $x$  and  $w$  variables in the set  $var(q)$  and fix their value at one

Else

Create  $x$  and  $w$  variables for each period in the restricted time windows

$T_{res}(s, q)$  and  $W_{res}(s, q)$

End If

End

Step 2: Solve the restricted problem [MP] model

Let  $QC$  be the set of trains whose time windows need to be extended by  $\delta$

For  $q \in Q$

If  $q$  is cancelled/delayed beyond planning horizon in the current solution

$QC := QC \cup \{ q \}$

$QC := QC \cup \{ q' : (q, q') \in QO \text{ or } (q', q) \in QO \}$

End If

End

Step 3: Time window relaxation

If  $QC = \emptyset$

Stop

Else

For  $q \in QC$

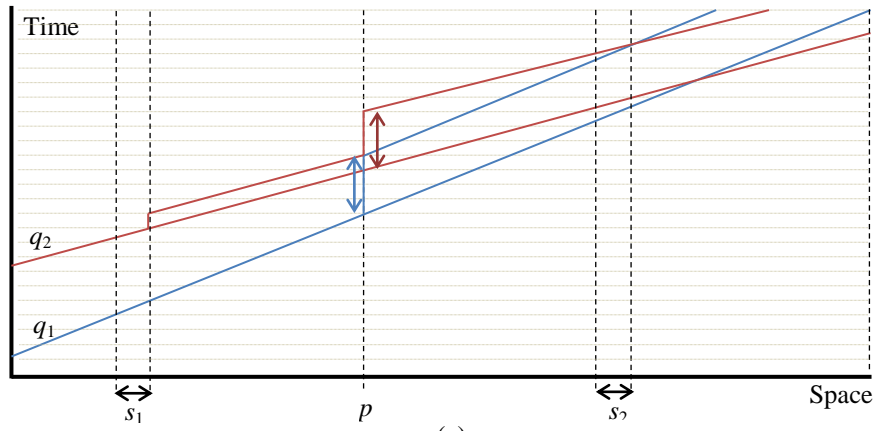
$\text{delay}(q) := \text{delay}(q) + \delta$

Update  $T_{res}(s, q)$  and  $W_{res}(s, q)$

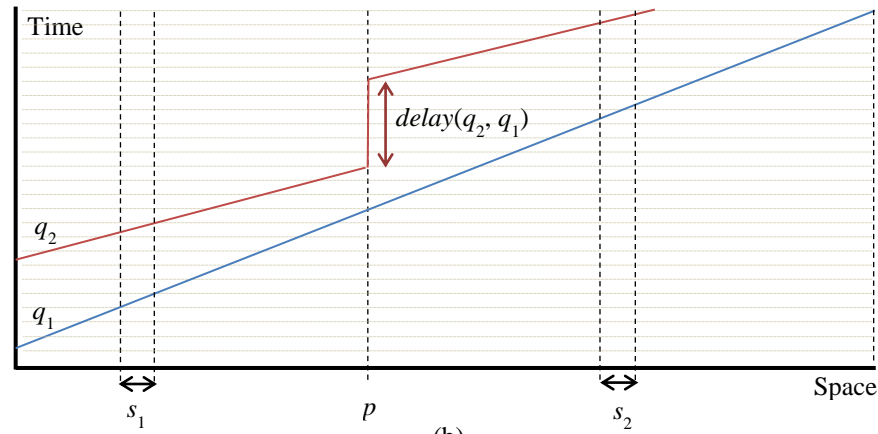
End For

Go to Step 1

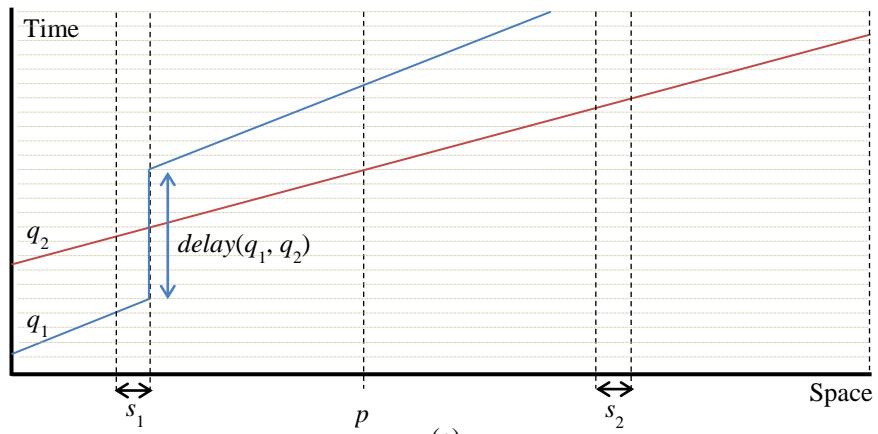
End



(a)



(b)



(c)

Figure 4.4 Illustration of LP-based delay estimation

(a) fractional paths (each fractional path has a weight of 0.5) (b)  $q_2$  follows  $q_1$  till end (c)  $q_1$  yields to  $q_2$  at siding  $s_1$

#### 4.5.2 Decomposition by train groups

In freight railways, trains are not released at equal-length time intervals, and their velocities can vary significantly. If we plot the unimpeded paths of all trains that are expected to enter a territory within a given planning horizon on a time-space network, we may see the paths cluster in groups. Each group consists of a few trains that are released one after another within a short time window, and travel in the same direction (maybe at different speeds). We propose a heuristic that solves conflicts between trains in one group in each iteration. Starting with the LP relaxation of the base model, we impose integrality requirements on the  $x$  and  $w$  variables for trains in a selected group, and solve a mixed-integer program (MIP). In the MIP solution, we fix all paths that are integral. In the next iteration, we select another group of trains whose MIP solution paths are fractional, impose integrality on their variables, and solve a new MIP problem. This procedure continues until all train paths are integral.

There are many possible ways to define train groups. We define train group as a set of trains traveling in the same direction and whose fractional paths in the LP solution overlaps (partial overtaking). In addition, we define all in-territory trains as one group and solve them in the first iteration. By doing this, we prevent the later MIP from becoming infeasible for in-territory trains (because we cannot “cancel” in-territory trains). We present a complete description of this algorithm below in procedure *train\_group\_decomposition*.

### ***Train\_group\_decomposition:***

Step 1: Solve the LP relaxation of the base model

Let  $QO$  be the set of train pairs whose fractional paths have partial overtaking in the LP solution

Let  $group(q)$  be the set of trains whose fractional paths have partial overtaking with train  $q$  and  $group\_QI$  be the set of all in-territory trains

For each train  $q \in Q$

    If train  $q$  has a unique and integral path in the LP solution

        Fix the  $x$  and  $w$  variables on the solution path of  $q$  to 1

    Else if  $q \in QI$

$group\_QI := group\_QI \cup \{q\}$

    Else

        Let  $group(q) := \{q\} \cup \{q' : (q, q') \in QO \text{ or } (q', q) \in QO\}$

    End If

End

Let  $G$  be the list of all train groups, i.e.,  $G := \{group(q) : q \in Q\} \cup \{group\_QI\}$

Remove small groups that are subsets of other train groups: if there exists

$group(q') \subseteq group(q)$ , remove  $group(q')$  from  $G$

Let  $group\_QI$  be the first group, then sort the other groups in decreasing order of the highest train priority in the group.

Step 2: Iteratively solve MIP

For each group  $g \in G$

    Impose integrality requirements on the  $x$  and  $w$  variables of each train in  $g$

    Solve MIP

    For each train  $q'$

        If train  $q'$  has a unique and integral path in the MIP solution

            Fix the  $x$  and  $w$  variables on the solution path of  $q'$  to 1

        End If

    End For

End

This train-group-based decomposition algorithm is most effective when the train groups are separated from each other and each group is relatively easy to solve. Otherwise, some iterations of MIP are difficult to solve.

#### 4.5.3 Rounding heuristic

The basic idea of our rounding heuristic is to first relax the integrality constraints on some or all variables and then iteratively round a variable with fractional value to 0 or 1. In our problem, a train's path splits into multiple fractional paths when it has partial waiting (fractional  $w$  variables) or partial detouring (fractional  $x$  variables entering parallel segments simultaneously). Instead of rounding the  $x$  or  $w$  variables directly, we introduce an auxiliary variable,  $z_p^{qt}$ , which equals 1 if train  $q$  passes control point  $p$  at or before time  $t$ , and 0 otherwise. Formally,  $z_p^{qt} = \sum_{t' \leq t} \sum_{s' \in B(p,q)} \sum_{s \in F(p,q)} x_{s's}^{qt'}$ , where  $B(p, q)$  and  $F(p, q)$  are the sets of backward and forward (in the direction of train  $q$ ) segments connected to control point  $p$ , respectively. Note that rounding a  $z$  variable is equivalent to imposing a time window on the train's entry to any downstream segments connected to control point  $p$ . If  $z_p^{qt} = 1$ , then train  $q$  must enter one of the segments in  $F(p, q)$  on or before  $t$ . If  $z_p^{qt} = 0$ , then the train needs to wait until time  $t$  before it enters downstream segments to control point  $p$ .

For this heuristic, we start with the LP relaxation of the base model plus the headway-based forcing constraints (4.13) and/or (4.14). As mentioned in Section 4.4.3, the headway-based forcing constraints can help to reduce the occurrence of fractional  $x$

variables caused by partial train detouring. This heuristic is an iterative algorithm where we select one fractional variable to round in each iteration. In any intermediate solution, if a train has integral path, we fix its entire path in the following iteration. There are inherently many possible ways to select a fractional variable and round it. Instead of simultaneously considering all fractional variables, we select one fractional variable for each train to compose a candidate set and then select one variable to be rounded from this set using different criteria. For a train with fractional paths in the solution, we may pick a  $z$  variable at the first control point where its path starts to become fractional. Among the  $z$  variables at the selected control point, we may consider options of pick the one with minimum, maximum, or average time index. After obtaining a set of candidate variables for rounding, one from each train with fractional  $z$  variables, we can select one of them based on certain criterion, such as (i) highest fraction value, (ii) minimum rounding error if the variable is rounded to the nearest integer, (iii) minimum time index of the variable, and (iv) highest train priority.

For a selected variable, we need to decide whether to round it to 0 or 1. One naïve way is to round it to the nearest integer. A more computationally expensive method is to evaluate the two options by solving the problem twice, where we round the variable to 0 in one run and 1 in the other. We then pick the rounding with lower objective value. A third method is a hybrid of the previous two options. Let  $[l_d, l_u]$  be a pre-determined range. We round the selected variable to 0 if its value is smaller than  $l_d$ , and 1 if its value is greater than  $l_u$ . If the fractional value falls between  $l_d$  and  $l_u$ , evaluate the two options by solving two new problems with one fixing the variable to 0

and the other fixing it to 1.

As we iteratively fix more and more  $z$  variables at 0 or 1, we continue to solve a more and more restricted problem. It is possible that we end up with an infeasible problem no matter how we round the next selected variable. Suppose the restricted problems are always feasible, then we will eventually obtain a solution where all  $z$  variables are integral, indicating that all  $w$  variables are integral. The algorithm terminates if all  $x$  variables are also integral. Otherwise, we select a fractional  $x$  variable and round it.

So far we have discussed three heuristic that start with solving the LP relaxation. In the next two sections, we propose two more heuristics that solve IP iteratively.

#### **4.5.4 Directional movement planning**

In this section, we propose a heuristic that iteratively solves subproblems that mainly consist of trains in one direction. The meet-pass plans in double-track territories can be complicated because trains traveling in both directions have the flexibility of moving between the parallel mainlines via crossovers. A simple traffic decomposition idea is to assign one mainline to trains in each direction and forbid the use of crossovers. Essentially, the double-track MP problem then becomes two single-track MP problems with one-way traffic. We can expect that the resulting solution will not fully utilize the track capacity that a double-track territory offers. The directional movement planning heuristic is based on the simple traffic decomposition idea, but is refined to enable the use of crossovers. The algorithm has two phases. First we solve two subproblems to construct a feasible solution, each subproblem solves for trains in one direction. Then,



we iteratively re-optimize the paths of a subset of trains to improve the initial solution.

To construct an initial solution, we select a direction  $d$ . The first subproblem consists of all trains traveling in direction  $d$ . We restrict these trains to use only the default mainline for direction  $d$  and the sidings connected to that mainline. The subproblem may also include in-territory trains that are initially on the selected mainline and sidings but travel in the opposite direction of  $d$ . We then solve this subproblem and fix the paths of all trains in direction  $d$ . In the next step, we solve the full MP problem with paths of trains in direction  $d$  fixed. Trains in the other direction are free to use any residual track capacity, including the default main line for direction  $d$ . This second step returns a feasible solution to the full movement planning problem.

Given an initial feasible solution, we continue with a local improvement procedure that iteratively re-optimizes paths of a subset of trains while fixing the other paths. Again, we will first re-optimize the paths of trains in direction  $d$ . But this time the subproblem also includes trains in the opposite direction of  $d$  which seem to be significantly delayed due to trains in direction  $d$ . The paths of the remaining trains are fixed as they are in the current solution. We then re-optimize the paths of the selected trains, allowing them to use any residual track capacity. This local improvement procedure is applied alternatively for the two directions until no further improvement is achieved.

Next we describe a method to estimate delay due to opposing traffic. Suppose we want to refine train paths in direction  $d$ . For a given train  $q$  in the opposite direction of  $d$ , let  $t_q$  be train  $q$ 's earliest arrival time at its ending segment via its unimpeded path,

assuming there are no other trains in the network. Then we fix the paths of all trains in direction  $d$  in the current feasible solution. Let  $t_q'$  be the arrival time at train  $q$ 's ending segment via the shortest path in the residual network where tracks are blocked for some periods due to the fixed routes. If  $t_q'$  is much later than  $t_q$  (say, 30 minutes later), we say that train  $q$  is *significantly delayed* due to trains in direction  $d$ . We use the same method to find all trains in the opposite direction of  $d$  that are significantly delayed and re-optimize their paths along with trains in direction  $d$ .

Finally, we note that this heuristic is not necessarily restricted to pure double-track territories. If some locations have more than two mainlines, we can pre-assign the default mainline(s) for each direction. However, if a portion of track is single track, then this method essentially gives priority to trains in the selected starting direction  $d$ . In this case, we may want to start with the direction with more high-priority trains.

#### 4.5.5 Expanding planning horizon

The last optimization-based heuristic we propose solves a sequence of subproblems with shorter planning horizon than the full problem. The goal is to solve each subproblem quickly and iteratively fix part of the solution and then extend the horizon until the subproblem covers the full horizon. Suppose we start with a horizon  $L_{sub} < L$  ( $L$  is the full horizon). The first iteration will solve for the movements from time 0 to  $L_{sub}$ . If this subproblem is solved within the given time limit, we will fix a portion of the train paths and erase the remaining movements. Then we expand the planning horizon  $L_{sub}$  and solve a new subproblem (with partial solution fixed). If the first subproblem is not solved, then we will reduce  $L_{sub}$  and resolve a smaller problem.

There are several parameters we can adjust to control the solution quality and computational time of this heuristic. First, we need to decide the initial planning horizon  $L_{sub}$ . One direct method is to set it to a pre-defined value based on experience. Here we define  $L_{sub}$  dynamically depending on the problem size and complexity. For example, we can count  $m$  trains or  $n$  passing conflicts in the unimpeded paths, whichever comes earlier, and let  $L_{sub}$  be the  $m^{\text{th}}$  train's release time or  $n^{\text{th}}$  conflict's time. Second, if a subproblem is solved, the amount of movements that we fix may also impact the algorithm's overall performance. To adjust this amount, we fix a percentage,  $p_{fix}$ , of the non-fixed movements in the current solution. Suppose we fixed all movements up to time  $L_{fix}$  in the last iteration, then we will fix the current solution up to time  $L_{fix} + p_{fix}(L_{sub} - L_{fix})$ . Then we can extend the horizon by counting another  $m$  trains or  $n$  passing conflicts starting from the updated value  $L_{fix}$ . Third, if a subproblem is not solved, we will reduce the horizon such that the new subproblem only solves a percentage of the original un-fixed horizon  $(L_{sub} - L_{fix})$ . To prevent the solution from being too myopic, we may also impose a lower bound on the horizon. For example, the new un-fixed horizon  $(L_{sub} - L_{fix})$  needs to have a minimum length. Further, the new horizon should be adequately larger than the horizon of the last solved subproblem. Finally, we set a short time limit on each subproblem. If a particular subproblem is not solved within the time limit and its size cannot be further reduced because its horizon is already at the minimum length, then we will remove the time limit and resolve the subproblem. A summary of this algorithm is presented as follows:

### ***Expand\_horizon:***

Definitions:

- $L$  = planning horizon of the full problem
- $L_{sub}$  = planning horizon of a subproblem
- $L_{solved}$  = planning horizon of the last solved subproblem
- $L_{fix}$  = horizon up to which train movements are fixed

Parameters:

- $\Delta t$  = minimum increment since  $L_{solved}$
- $t_{min}$  = minimum increment since  $L_{fix}$
- $t_{limit}$  = time limit for a subproblem
- $p_{fix}$  = percentage of the movements to be fixed in the next iteration
- $p_{reduce}$  = new planning horizon expressed as a percentage of the horizon of the current subproblem

Initialization:

$$L_{solved} = 0, L_{fix} = 0,$$

$$L_{sub} = \text{time of the } m^{\text{th}} \text{ train's release time or } n^{\text{th}} \text{ conflict's time, starting from } L_{fix}$$

Step 1: Solve a subproblem with a planning horizon of  $L_{sub}$ .

If the subproblem has the same horizon as the last (unsolved) subproblem, do not impose time limit; otherwise, set a time limit of  $t_{limit}$  on the solver for the subproblem. If the subproblem is solved, go to Step 2; otherwise, go to Step 3.

Step 2: Partially fix the solution and expand the horizon.

If  $L_{sub} = L$ , stop.

Otherwise, let  $L_{fix} := L_{fix} + (L_{sub} - L_{fix}) * p_{fix}$ . Fix new movements up to time  $L_{fix}$ .

Let  $L_{solved} := L_{sub}$ .

Let  $L_{sub}$  be the time of the  $m^{\text{th}}$  train's release time or  $n^{\text{th}}$  conflict's time starting from  $L_{fix}$ , or  $L$ , whichever is smallest. Go to Step 1.

Step 3: Reduce planning horizon.

Let  $L_{sub} := \max\{ L_{fix} + (L_{sub} - L_{fix}) * p_{reduce}, L_{fix} + t_{min}, L_{solved} + \Delta t \}$ . Go to Step 1.

So far we have discussed 5 heuristics, all of which use the optimization model [MP] and/or its relaxation, and rely on LP/MIP solver. In the next section, we propose a

standalone heuristic that is completely independent of the optimization model and solver.

#### 4.6 SEQUENTIAL CONFLICT RESOLUTION HEURISTIC

The optimization-based heuristics discussed in Section 4.5 generally cannot find feasible solutions quickly (e.g., within a few minutes), especially for large problem instances. In this section, we propose a standalone heuristic that can generate good solutions very quickly. The algorithm has two stages: initial solution construction and local improvement. In the first stage, we find train paths by one movement at a time and resolve conflicts as they occur. The second stage then applies a single-train re-routing algorithm to improve the initial solution. In the following sub-sections, we first present the core algorithm of Stage 1, and then discuss three variations that use different methods to process the trains and select meet-pass plans for each conflict.

##### 4.6.1 Construct an initial solution

In Stage 1, we advance one train to the next segment in each iteration, and resolve the conflicts that may arise between the new movement and existing movements of other trains. Throughout this discuss, we use *move* to refer to the movement of a train on a segment with specified entry time and exit time. An *active move* is the last move on a current train path, representing the current location and ready time of the train. We put all active moves in a list (unless it is a move to the train’s ending segment or its exit time is beyond the planning horizon), and sort in increasing order of the exit time of the moves. Figure 4.5 outlines the major steps in constructing an initial solution, including *initialization*, *advancing*, *conflict detection*, *conflict resolution*, *backtracking*, and *updating*. To start the path-construction process, we initialize the active move list with

one initial move for each train, representing the train's starting segment and release time. In each iteration, we pick a move from the list and temporarily move the *active train* to the next segment along the train's preferred path. Then, the conflict detection step will check for violations of section capacity of the new move's segment and headway requirement at the control point between the active move and the new move. If the new move does not conflict with any existing move, we simply append the new move to the train's path and update the list of active moves. Otherwise, we apply several conflict resolving strategies to find a meet-pass plan between two trains in conflict. These strategies are localized so that one of the two trains in conflict will wait in a previous segment or switch to a parallel track to avoid the conflict. We may consider backtracking one of the two trains further upstream if all local strategies fail. If the conflict resolution routine finds a set of moves that can resolve the conflict, then we can update the train paths and active moves, and proceed to the next iteration. Otherwise, the heuristic procedure fails to find a feasible movement plan and stops. We repeat this procedure until all trains are at their destinations or the end time of a train's active move exceeds the planning horizon.

We discuss the details of conflict detection and strategies to resolve a conflict in the procedures *conflict\_identification* and *solve\_conflict*, respectively. For this purpose, we use  $m$  to denote a move and describe it with a tuple  $(q, s, a, b)$ , representing train  $q$  entering segment  $s$  at time  $a$  and exiting from it at time  $b$  ( $b - a \geq \tau_{sq}$ ). Recall that  $\tau_{sq}$  is the unimpeded travel time of train  $q$  on segment  $s$ . With some abuse of notation, we use  $\tau_{nq}$  to represent the unimpeded travel time of train  $q$  through section  $n$ ,

and let  $\sigma_{nq}$  be the travel time from the entry point of segment  $s$  to the entry point of section  $n$ , for all  $n \in N(s)$ . Thus, given a move  $m(q, s, a, b)$ , train  $q$  occupies section  $n$  during  $[a+\sigma_{nq}, a+\sigma_{nq}+\tau_{nq})$  if  $n$  is not the last section of segment  $s$  ( $\delta_{nq} = 0$ ), and  $[a+\sigma_{nq}, b)$  otherwise ( $\delta_{nq} = 1$ ). Finally, let  $lst(m)$  be the last move before  $m$  along train  $q$ 's path.

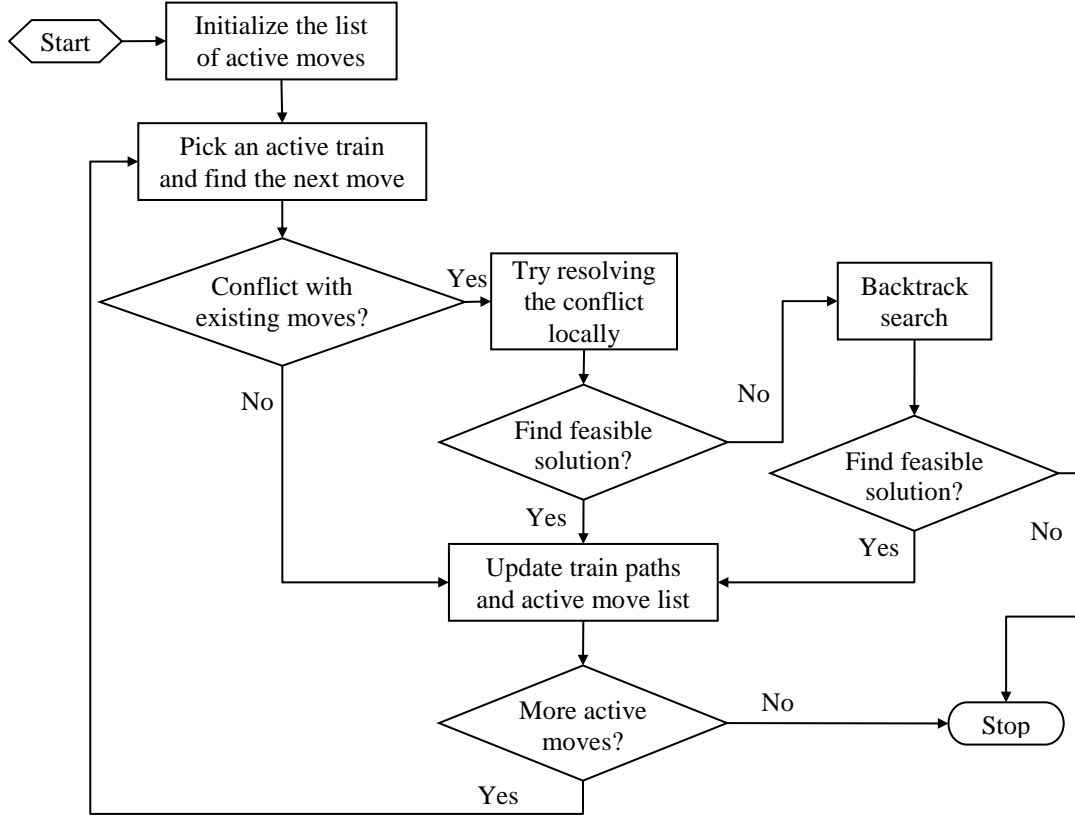


Figure 4.5 Construct initial solution

***Conflict\_identification:***

Let  $m(q, s, a, b)$  be the new move to be checked. Find the earliest existing move,  $m'(q', s, a', b')$ , that is in conflict with  $m(q, s, a, b)$ .

Step 1: Check for overtaking conflict

An existing move  $m'(q', s, a', b')$  has an overtaking conflict with  $m(q, s, a, b)$  if:

- (i)  $q'$  travels in the same direction as  $q$ , and
- (ii) the two moves' start and end times satisfy one of the following conditions:
  - $[a + \sigma_{nq}, a + \sigma_{nq} + \tau_{nq}) \cap [a' + \sigma_{nq'}, a' + \sigma_{nq'} + \tau_{nq'}) \neq \emptyset$ , for any  $n \in N(s)$  and  $\delta_{nq} = 0$
  - $[a + \sigma_{nq}, b) \cap [a' + \sigma_{nq'}, b') \neq \emptyset$ , for  $n \in N(s)$  and  $\delta_{nq} = 1$
  - $|a - a'| < \mu$
  - $|b - b'| < \mu$

Step 2: Check for meeting conflict

An existing move  $m'(q', s, a', b')$  has a meeting conflict with  $m$  if:

- (i)  $q'$  travels in the opposite direction of  $q$ , and
- (ii)  $a' < b + \mu$  and  $a < b' + \mu$

Step 3: Check for headway conflict

An existing move pair  $lst(m') \rightarrow m'$  has a headway conflict with  $lst(m) \rightarrow m$  if:

- (i)  $lst(m) \rightarrow m$  and  $lst(m') \rightarrow m'$  use segment pairs belonging to the same headway set. If  $g, s, g', s'$  are segments of  $lst(m), m, lst(m'),$  and  $m'$ , respectively, and  $p$  is the control point connecting  $g$  and  $s$  (or  $g'$  and  $s'$ ), then  $(g, s) \in H_i(p)$  and  $(g', s') \in H_i(p)$  for some  $i \in \{1, 2, h_p\}$ , and
- (ii)  $|a - a'| < \mu$ .

If the conflict detection routine finds multiple moves that are in conflict with the new move of the active train, we will only pick the move with the smallest starting time. Then, we will try to use the local conflict resolving strategies to solve the conflict. Note that the solution moves returned by any of these strategies may incur new conflicts. When such new conflicts arise, we may either deny the solution and try other strategies or backtrack the moves in conflict with the solution moves by one step. We may favor one option over the other depending on the system state. In general, we want to keep trains traveling in the same direction on the same mainline when conflicts arise. Therefore, if the solution suggests one of the two trains currently in conflict should move to a parallel track and it incurs new meeting conflicts, we will deny the solution and try other



strategies. If all local strategies fail to find a feasible solution to the conflict, we will backtrack one of the two trains in conflict further to an upstream location where it can wait and let it stay there till its unimpeded path downstream can avoid the current conflict. One potential problem with our conflict resolving strategies is looping. In some involved cases, several trains conflict each other and backtracking may take us to the same system states over and over again, hence, an infinite loop. In this case, the procedure simply aborts after it reaches the maximum number of iterations.

### ***Solve\_conflict:***

Suppose two moves,  $m_1(q_1, s_1, a_1, b_1)$  and  $m_2(q_2, s_2, a_2, b_2)$ , are in conflict.

- Apply strategies to solve an overtaking conflict if  $s_1 = s_2$  and  $q_1, q_2$  travel in the same direction
- Apply strategies to solve a meeting conflict if  $s_1 = s_2$  and  $q_1, q_2$  travel in opposite directions
- Apply strategies to solve a headway conflict if  $s_1 \neq s_2$

### **Solve overtaking conflict**

Assume  $a_1 \leq a_2$ .  $lst(m_1)$  and  $lst(m_2)$  use segments  $g_1$  and  $g_2$ , respectively ( $g_1$  and  $g_2$  may or may not be the same).

- Extend the waiting time of  $m_2$   
Let  $q_2$  wait on  $s$  until time  $b_1 + \mu$  if  $m_1$  and  $m_2$  do not overtake but  $b_2 < b_1 + \mu$ .
- Extend the waiting time of  $lst(m_1)$  or  $lst(m_2)$   
Let  $q_1$  wait on  $g_1$  until time  $a_1'$  if  $g_1 \neq g_2$ , or let  $q_2$  wait on  $g_2$  until time  $a_2'$ .  
Time  $a_1'$  (or  $a_2'$ ) is computed such that the unimpeded downstream move  $m_1'(q_1, s, a_1', a_1' + \tau_{sq_1})$  (or  $m_2'(q_2, s, a_2', a_2' + \tau_{sq_2})$ ) do not conflict with  $m_2$  (or  $m_1$ ).
- Replace  $m_1$  or  $m_2$  with a move on a parallel segment of  $s$ :  $s'$   
Let  $q_1$  take an alternative move  $m_1'(q_1, s', a_1, b_1')$  or let  $q_2$  take an alternative move  $m_2'(q_2, s', a_2, b_2')$ .  
Time  $b_1'$  or  $b_2'$  is the earliest time when an immediate downstream segment of  $s'$  becomes available (i.e., not blocked by any existing moves).
- Replace  $lst(m_1)$  or  $lst(m_2)$  with a move on a parallel segment of  $g_1$  or  $g_2$

Let  $q_1$  or  $q_2$  switch to a parallel track in the previous step, i.e., replace  $lst(m_1)$  with its alternative move  $lst'(m_1)$  on segment  $g_1'$ , or replace  $lst(m_2)$  with its alternative move  $lst'(m_2)$  on segment  $g_2'$ .

Update the exit time of  $lst'(m_1)$  or  $lst'(m_2)$  to be the earliest time when an immediate downstream segment of  $g_1'$  or  $g_2'$  becomes available.

### Solve meeting conflict

Assume we designate  $q_1$  to yield to  $q_2$ .

- Replace  $m_1$  with a move on a parallel segment of  $s$ :  $s'$   
Let  $q_1$  take an alternative move  $m_1'(q_1, s', a_1', b_1')$ .  
Update  $b_2$  such that  $b_2' = a_1 + \mu$  and  $a_1' = a_1$ , or set  $a_1' = b_2 + \mu$ .  
Time  $b_1'$  is the earliest time when an immediate downstream segment of  $s'$  becomes available.
- Replace  $lst(m_1)$  with a parallel move  $lst'(m_1)$  on segment of  $g_1'$   
Let  $q_1$  switch to a parallel track at the previous control point if it cannot switch from  $g_1$  to  $s'$ .  
Update the exit time of  $lst'(m_1)$  to be the earliest time when an immediate downstream segment of  $g_1'$  becomes available.
- Extend the waiting time of  $lst(m_1)$   
Let  $q_1$  wait on  $g_1$  until an immediate downstream segment ( $s$  or  $s'$ ) becomes available if this does not prevent  $q_2$  from going forward from  $s$ .

### Solve headway conflict

Let  $q_1$  wait on  $g_1$  until  $a_1' = a_2 + \mu$  or let  $q_2$  wait on  $g_2$  until  $a_2' = a_1 + \mu$ .

In the construction of initial solution, there are several decisions we need to make when solving a meet/pass conflict, including how to select the active train in each iteration, how to determine the yielding train in a conflict, which train to be backtracked when local conflict resolution fails, and how to pick a meet-pass plan when multiple solutions are feasible. We developed three variations, *rule-based*, *lookahead*, and *randomized*, that implement these decisions differently. The rule-based method applies prescribed rules to make the decisions. Each iteration starts with picking the first active

move in the list, i.e., the active move with minimum exit time, and moves the active train forward. If the new move has an overtaking conflict with an existing move, we will first decide whether overtaking should happen, based on the trains' relative priority and velocity. If we want the following train to overtake the leading train, then the leading train needs to wait or detour so as to yield to the following train. Otherwise, the following train will keep following or detour. For meeting conflicts, we don't predetermine the yielding train. Instead, we will compare the local cost of two plans: train  $q_1$  yields to  $q_2$  vs.  $q_2$  yields to  $q_1$ , and choose the solution with lower cost. If local conflict resolution fails, then we will backtrack the train with lower priority involved in the conflict. For all these decisions, if our prescribed decision fails to find a feasible solution to solve a conflict, we will try the alternative pass-yield decision. For instance, if we cannot find a solution when we choose to let the following train overtakes, then we will try the option of no overtaking.

The lookahead method builds upon the rule-based method. The key difference is that we do not predetermine the yielding train in any meet/pass conflict. Instead, whenever there is a conflict between two trains,  $q_1$  and  $q_2$ , we try both options of letting train  $q_1$  yields to  $q_2$  or letting  $q_2$  yields to  $q_1$ . For a given decision, we solve the current conflict with the designated yielding train. Then, we keep planning for the remaining horizon for a specified number of hours using the rule-based method, i.e., look ahead to see the future meet-pass plans of all trains. After obtaining two (incomplete) solutions, we then compare their objective values and fix the pass-yield decision as it is in the

solution with lower cost. This lookahead method is computationally more intensive, but it usually makes better pass-yield decisions by looking into future meet-pass plans.

The last variation, randomized decision making, is proposed by Balakrishnan et al. (2012). In our implementation, we add certain level of randomness in several decisions, including active train selection, overtaking decision, meeting plan selection, and backtracking decision. When making these decisions, we first assign the probability that each option will be selected and then generate a random number to determine the selection of an option. First, when selecting the active train of each iteration, we may randomly (with equal probabilities) select an active move among moves that have similar exiting times. Second, if two train movements have an overtaking conflict, the probability that we will let the following train overtake the leading train at that location depends on their relative priorities and velocities. Third, we will generate multiple meeting plans for a meeting conflict (using the strategies described in procedure *solve\_conflict* and randomly select a plan, with the cheapest solution being selected with higher probability. Finally, the backtracking step is more likely to backtrack a train with lower priority in a meeting conflict. For overtaking conflicts, backtracking decision is the same as the overtaking decision. That is, we will backtrack the following train if we do not want it to overtake or the leading train otherwise.

This randomization method allows us to run the heuristic multiple times and generate different solutions. Randomization may yield better solution than the deterministic rule-based and lookahead methods because local optimal decisions do not necessarily lead to near-optimal overall solution. Further, when the deterministic

methods fail, we may still be able to find feasible solutions using randomization. We can execute these heuristic methods in parallel on a computer with multiple CPU cores and select the solution with the minimum objective value.

#### **4.6.2 Train re-routing**

Given an initial feasible solution, we can apply a simply re-routing algorithm to improve the solution. The idea is to find a shortest path for a selected train assuming that the paths of all other trains are fixed. To apply the shortest path algorithm, we construct a time-space network where each node represents track and time resource and each arc represents a movement or waiting period of the train selected for re-routing. Then we remove a subset of the arcs to forbid infeasible train movements and waiting due to the fixed movements of other trains. The residual time-space network is directed and acyclic, we can find the shortest path of the selected train very efficiently using topological sorting.

To start the algorithm, we select all trains with positive delay in the current solution and sort them in certain order in a list. Then we iteratively re-route the trains in the list, one at a time, until no more improvement is achieved. Note that we may need to go through the list multiple times because finding a new path for a given train may provide improvement opportunities for other trains. Further, we may get different final solutions if we re-route the trains in different order. For instance, we may apply re-routing in the order of increasing train priority, amount of delay in the current solution, or trains' order of entry to the territory.

This re-routing algorithm can be applied to any feasible solution. In the computational experiments of Section 4.7, we will use different heuristic methods to obtain an initial solution and then improve it using this train re-routing procedure.

#### **4.7 COMPUTATIONAL RESULTS**

In this section, we construct real-size problem instances to test the base [MP] model and the effectiveness of various heuristic methods. Specifically, we will examine three heuristics based on LP relaxation (time window method, train group decomposition, and rounding heuristic), directional MP, expanding horizon method, and the standalone conflict resolution heuristic. The performance of these methods is compared to solving the base model using CPLEX MIP solver.

The territory is a stretch of 145-mile long double tracks with 1 siding attached to main 1 and 3 sidings attached to main 2. There are 10 crossovers that allow trains to move from one main to the other. To measure the performance of our solution methods in real life, we generated problem instances that mimic the real-life traffic pattern in a double-track territory. Each problem instance has over 20 trains in a 6-hour horizon, with a mixture of direction, priority, unimpeded velocity, and entry time to the territory. Train priority is randomly generated in the range of 1 to 11, and train velocity varies in the range of 40 to 80 miles per hour. In general, train's velocity is larger when it has higher priority. Assume the default main of eastbound trains is main 1 and that of westbound trains is main 2.

Our computation tests were conducted on a host with Intel(R) Xeon(TM) 3.73GHz CPUs and 24 GB shared memory. We used ILOG CPLEX 12.4 to solve the

LP, MIP, and IP models. We set CPLEX to use deterministic parallel search, using up to 24 threads on the server. The stopping criteria were 1% optimality gap and a time limit of 2 hours, whichever happened earlier. If a method is an iterative algorithm, for each iteration CPLEX is given a time limit of remaining time up to 2 hours.

The testing pool consists of 20 randomly selected instances, among which 10 (labeled as E01, E02, ..., E10) were solved to optimality within 2 hours using the base model and the other 10 (labeled as H01, H02, ..., H10) were not. To compare the effectiveness of the methods, we measure the following aspects: (1) model size in terms of number of variables and constraints; (2) objective value; (3) solution gap between the best solution and best lower bound, expressed as a percentage of the best solution; (4) solution time, including CPLEX solving time and total time. Total time includes CPLEX solving time and time needed for all other operations such as constructing the problem instance, preparing CPLEX model, and solution visualization in a time-space network. For each heuristic algorithm, we apply the re-routing procedure of Section 4.6.2 to refine the final solution obtained. We use INF to indicate an infeasible instance, and NFS to indicate that CPLEX failed to find any feasible solution within the time limit.

Instance	# train	# x var	# w var	# v var	# depart constr	# flow constr	# headway constr	# section capacity constr	# train-based forcing constr
E01	20	155,315	88,957	16,790	20	88,959	15,603	71,512	58,684
E02	26	200,550	115,356	17,186	26	115,360	16,308	75,793	74,587
E03	26	169,325	97,566	17,126	26	97,564	16,098	75,048	63,978
E04	24	233,751	134,800	17,910	24	134,806	16,905	79,056	87,900
E05	24	180,752	103,487	16,802	24	103,499	15,687	71,546	67,505
E06	21	157,374	90,416	17,178	21	90,427	16,236	72,155	57,674
E07	20	188,954	110,069	17,754	20	110,066	16,539	70,214	71,928
E08	25	203,927	118,140	17,870	25	118,146	16,862	78,507	76,120
E09	23	204,736	118,458	17,352	23	118,462	16,404	73,789	76,092
E10	24	141,402	80,930	17,712	24	80,935	16,561	74,435	53,739
H01	31	249,090	145,378	18,062	31	145,381	17,024	76,207	92,512
H02	27	211,511	122,385	18,182	27	122,390	16,964	77,154	79,772
H03	21	199,022	114,420	16,900	21	114,426	15,847	73,534	74,122
H04	28	240,870	140,314	18,014	28	140,314	16,992	79,385	89,362
H05	28	291,761	169,063	18,110	28	169,065	16,977	80,312	108,724
H06	20	206,085	119,372	17,882	20	119,375	16,883	72,581	76,081
H07	25	200,604	115,841	17,196	25	115,840	16,446	73,202	72,956
H08	31	258,691	148,724	17,486	31	148,733	16,385	61,908	97,134
H09	27	267,383	154,365	17,714	27	154,372	16,668	78,163	99,319
H10	24	217,669	125,810	17,758	24	125,823	16,729	78,640	82,199

Table 4.1 Base model problem size and model size

#### 4.7.1 Base Model

To provide a benchmark for performance comparison, we first solve the 20 selected instances using the base [MP] model (4.1) – (4.12). Table 4.1 summarizes the sizes of these instances. Each instance has 20 to 31 trains, with a mixture of in-territory trains and future trains travelling in two directions. With a 6-hour planning horizon, the



base model has over 240,000 variables, most of which are  $x$  variables. The number of constraints varies in the range of 225,000 to 375,000. Table 4.2 summarizes the computational results of the base model. The first 10 instances, E01 to E10, are relative easy and are solved to optimality ( $\text{gap} \leq 1\%$ ) within the 2-hour time limit. The other 10 instances terminated with a gap of 23% to 95.5%. In general, the easy problems have fewer variables and constraints. But problem size is not the sole factor that determines computational complexity. For example, instance H06 only has 20 trains but CPLEX only found a solution with 52% gap in 2 hours. One important factor that impact solution time is the number and time/location distribution of conflicts in the unimpeded paths. Table 4.2 also lists the time CPLEX spent to solve root relaxation and the number of B&B nodes explored. These two statistics seem to be closely related to model size: for problems with larger size CPLEX tends to spend more time solving a node problem and explore fewer nodes within the time limit.

In the following sections, we discuss the computational performance of each of the heuristic algorithms we proposed in section 4.5 and 4.6 compared to the base model.

#### **4.7.2 LP-based heuristics**

In this section, we discuss the three heuristics that start with solving an LP relaxation, namely, LP-based time window method, decomposition by train group, and rounding heuristic.

Instance	Soln status	Obj	Lower bound	Soln gap	Root node soln time (sec)	# B&B nodes	Total time (min)
E01	Optimal	307	306	0.2%	15	3,737	16
E02	Optimal	345	342	0.9%	37	903	23
E03	Optimal	508	505	0.6%	14	5,287	27
E04	Optimal	314	311	1.0%	23	1,046	30
E05	Optimal	365	364	0.1%	16	512	32
E06	Optimal	479	474	1.0%	12	4,059	52
E07	Optimal	545	540	1.0%	10	14,686	55
E08	Optimal	570	565	1.0%	22	1,646	70
E09	Optimal	441	437	1.0%	38	2,842	94
E10	Optimal	560	554	1.0%	53	4,383	107
H01	Feasible	1,396	1,074	23.0%	234	127	122
H02	Feasible	634	467	26.3%	26	2,147	120
H03	Feasible	558	346	38.0%	16	6,260	126
H04	Feasible	989	520	47.4%	38	811	120
H05	Feasible	1,485	763	48.6%	202	40	120
H06	Feasible	826	398	51.8%	21	2,383	120
H07	Feasible	3,247	776	76.1%	57	120	120
H08	Feasible	7,676	838	89.1%	182	50	120
H09	Feasible	10,275	650	93.7%	133	181	120
H10	Feasible	12,393	560	95.5%	63	339	120

Table 4.2 Base model computation result summary

### LP-based time window

The time window heuristic aims at reducing model size by imposing an estimated delay time window on each train. The initial time windows are derived from the fractional train paths in the LP solution. If the restricted IP problem is infeasible, we extend the time windows of a subset of trains by 10 minutes and resolve. Table 4.3 shows that most LP relaxations of the base model were solved within 1 minute and up to

5 minutes for all remaining instances. There were 0 to 6 trains with integral paths in the LP solution. These integral paths were fixed before solving the restricted IP. The trains with fractional paths had an average time window of only 5 to 12 minutes. These time windows were quite tight and helped to significantly reduce the model size. As we can see from Table 4.4, the restricted problems only had 3% to 9% of the  $x$  and  $w$  variables and 14% to 23% of the constraints of the base model. The model sizes increased slightly in the second iteration (7% to 13%  $x$  and  $w$  variables, 21% to 28% constraints) as we relax some trains' time windows. The algorithm terminated within two iterations for all instances, either with a feasible solution or due to time limit (2 hours). After the first iteration, 9 out of the 20 instances obtained a feasible solution and the algorithm terminated. The remaining 11 instances had 1 to 4 trains cancelled or delayed beyond the horizon and we ended up with extending the time windows of 7 to 24 trains in the second iteration (see the last two columns of Table 4.3). Among these 11 instances, 4 of them (H05, H08, H09, and H10) were terminated due to the 2-hour time limit (shown as NFS in Table 4.4); the first iteration of these instances did not solve to optimality due to time limit. Notice that the final solutions of these 4 instances had large objective values since one or more trains were cancelled or pushed out of the planning horizon. The other 7 instances obtained a feasible solution in the second iteration. The final objective value shown in Table 4.4 was obtained by applying re-routing procedure on top of the terminating solution of time window heuristic.

The time window heuristic significantly reduced computational time for all 10 easy instances (compared to the based model). For the hard instances, although this

algorithm still reached the 2 hour time limit for 6 out of 10 instances, but the final solutions were better than the those of the base model in all of them except H05.

Overall, this heuristic is quite effective.

Instance	LP soln time (sec)	# train with time window	Avg. time window (min)	# integral path	# train cancelled in iter 1	# train extended time window in iter 2
E01	24	14	7.6	6	0	0
E02	30	26	5.5	0	1	7
E03	6	23	8.3	3	0	0
E04	6	23	4.8	2	4	23
E05	24	22	5.6	2	1	9
E06	6	20	9.4	2	2	7
E07	6	15	11.5	6	0	0
E08	12	23	7.2	2	0	0
E09	36	20	12.4	3	0	0
E10	42	21	10	3	0	0
H01	288	26	10.2	5	1	20
H02	48	21	6.1	6	0	0
H03	12	19	9.1	2	0	0
H04	60	28	5	0	1	18
H05	120	27	8.5	1	2	18
H06	6	18	10.6	2	0	0
H07	48	25	5.7	0	3	12
H08	174	25	11	6	3	24
H09	318	26	7.8	1	1	13
H10	36	22	8.5	2	1	7

Table 4.3 LP-based time window heuristic: time window and integral path

Instance	Restricted [MP] iter 1					Restricted [MP] iter 2					Final Obj	Total time (min)
	% x, w var	% constr	Obj	Soln gap	Cplex time (min)	% x, w var	% constr	Obj	Soln gap	Cplex time (min)		
<b>E01</b>	4%	15%	308	0.9%	2						308	3
<b>E02</b>	5%	17%	2,796	0.4%	1	7%	21%	345	0.9%	3	345	5
<b>E03</b>	6%	17%	508	1.0%	2						508	3
<b>E04</b>	3%	14%	16,563	0.1%	0	9%	23%	314	0.9%	3	314	3
<b>E05</b>	4%	17%	1,907	0.8%	6	7%	22%	365	0.2%	2	365	9
<b>E06</b>	8%	23%	1,418	0.5%	3	10%	26%	712	1.0%	18	712	21
<b>E07</b>	6%	18%	560	1.0%	3						560	4
<b>E08</b>	5%	18%	600	1.0%	1						600	2
<b>E09</b>	8%	22%	453	1.0%	10						453	11
<b>E10</b>	9%	23%	560	0.9%	11						560	12
<b>H01</b>	6%	18%	3,497	0.5%	2	10%	23%	1,109	1.0%	42	1,109	50
<b>H02</b>	4%	15%	560	0.9%	48						560	50
<b>H03</b>	7%	20%	481	1.0%	44						481	45
<b>H04</b>	3%	14%	3,216	1.0%	25	7%	21%	741	6.7%	93	741	120
<b>H05</b>	5%	17%	8,344	1.7%	118	9%	24%	NFS	--	--	8,344	122
<b>H06</b>	7%	19%	732	0.9%	2						732	2
<b>H07</b>	5%	16%	11,739	1.0%	1	8%	21%	1,011	18.3%	118	996	120
<b>H08</b>	8%	21%	6,092	54.8%	116	13%	28%	NFS	--	--	5,271	123
<b>H09</b>	5%	17%	5,921	88.7%	112	8%	21%	NFS	--	--	5,891	121
<b>H10</b>	6%	18%	793	4.5%	119	8%	21%	NFS	--	--	793	120

Table 4.4 LP-based time window heuristic: model size and computation results

### Decomposition by train groups

The second LP-based heuristic starts with solving an LP relaxation, and then iteratively fixes integral train paths in the previous solution and imposes integrality requirements on the variables of a group of trains. Table 4.5 presents the sizes of train groups and computational time in each iteration. As we can see from the last two

columns, train group decomposition algorithm obtained optimal or near-optimal solutions for all easy instances. The overall computational time was reduced for 7 out of the 10 instances but increased for the other 3. The average computational time reduced from 51 to 37 minutes. However, this heuristic did not perform well for the hard instances. It only solved 2 instances within the time limit and obtained solutions that were better than those of the base model. The other 8 instances terminated due to time limit and their final solutions were still fractional for some trains. In fact, 6 of these 8 instances even did not solve the first iteration in 2 hours (recall that the first iteration imposes integrality constraints on all in-territory trains).

For the 12 instances that obtained a feasible solution, there were typically more than 3 train groups in the problem and thus the algorithm solved for at least 3 iterations (except for E10). Each iteration imposed integrality on variables of 1 to 5 trains, averaged at 1.4 to 3 trains. Note that the columns of train group size do not count trains whose paths are integral and fixed. Also, we only recorded the group sizes for the iterations that were solved (for instance, H01 has more than one train group, but we only had the size for the first group because the algorithm did not even start to solve the second group). Although the train group size does not vary much from iteration to iteration for a particular instance, but the time CPLEX needed to solve the MIP varies widely from 1 minute to 2 hours. In general, CPLEX computation time is positively correlated to the size of train group, but it is not always so. For example, H03 only has 2 trains in the first iteration, but it did not solve in 2 hours.

Instance	Train group size each iter			Cplex time each iter (min)			# MIP iter	Final Obj	Total time (min)
	Min	Max	Avg	Min	Max	Avg			
<b>E01</b>	1	3	2.0	1	5	4	3	307	13
<b>E02</b>	1	4	2.0	1	9	4	5	345	22
<b>E03</b>	1	3	2.3	0	15	5	7	686	34
<b>E04</b>	2	5	3.0	3	4	3	3	314	11
<b>E05</b>	1	2	1.4	1	13	3	8	365	26
<b>E06</b>	2	3	2.3	1	57	16	4	485	63
<b>E07</b>	1	4	2.5	2	8	5	4	561	21
<b>E08</b>	1	5	2.2	1	11	4	6	570	25
<b>E09</b>	1	3	2.2	1	11	5	6	441	30
<b>E10</b>	2	4	3.0	2	117	60	2	560	120
<b>H01</b>	8	8	8.0	115	115	115	1	NFS	123
<b>H02</b>	1	3	2.0	1	13	4	5	571	24
<b>H03</b>	2	2	2.0	120	120	120	1	NFS	121
<b>H04</b>	9	9	9.0	119	119	119	1	NFS	121
<b>H05</b>	6	6	6.0	118	118	118	1	NFS	122
<b>H06</b>	1	5	2.5	1	100	20	6	728	120
<b>H07</b>	1	5	2.3	4	107	39	3	NFS	120
<b>H08</b>	4	4	4.0	116	116	116	1	NFS	120
<b>H09</b>	5	5	5.0	115	115	115	1	NFS	120
<b>H10</b>	2	3	2.5	49	70	60	2	NFS	120

Table 4.5 Characteristics and performance of decomposition by train groups

### Rounding heuristic

The rounding heuristic iteratively solves LP; each iteration fixes integral train paths and adds a constraint to fix one fractional variable at either 0 or 1. For this heuristic, we consider adding the headway-based forcing constraints discussed in Section 4.4.3 to eliminate some solutions with fractional  $x$  variable and potentially get more integral train paths in the LP relaxation solution. In particular, we consider the following three options: (1) base model + constraint (4.13), (2) base model + constraint

(4.14), and (3) base model + constraints (4.13) and (4.14). We compare the LP solutions of these 3 models to that of the base model and present the results in Table 4.6. The numbers of constraints (4.13) and (4.14) are similar, both in the range of 12,000 to 18,000. Adding either one of these two forcing constraints can improve the LP objective value up to over 50% (expressed as a percentage of the objective value of the base model LP). Option (3), which adds both forcing constraints, can improve the LP bound by 88% in some instances. Also, adding these forcing constraints helps to obtain more integral train paths in the LP solution, especially when we add both (4.13) and (4.14). While we can obtain tighter LP bound and more integral paths, the expanded models do took longer time to solve than the base LP. For option (1), only H08 took 5 more minutes to solve. But for options (2) and (3), there were 3 and 5 instances, respectively, that took significantly longer time to solve (more than 5 minutes). For the rounding heuristic, we prefer a tight model that can generate more integral paths in the LP solution (so we can fix more paths in the first iteration). Also, we would like the solution time to be short because we are likely to solve many iterations in this algorithm. Taking into account both goals, we use option (1), base model plus constraint (4.13), for the rounding heuristic.

Given a fractional solution, there are numerous ways to select the variable and round it. Using 14 small-size instances with 14 to 29 trains in a 6-hour horizon, we tested 36 different settings. Each setting chooses one option from each of the following categories: (1) variable location: first control point/segment or last control point/segment; (2) time index of  $z$  variable at the selected location: last period or middle period; (3)



select a variable from candidate pool: maximum fractional value, minimum rounding error to the nearest integer, or smallest time index; (4) rounding direction: round to the nearest integer, evaluate both rounding directions, or a hybrid method that rounds to the nearest integer if the fractional value is outside a given range  $[l_d, l_u]$  and evaluate both rounding directions otherwise. Our testing result showed that some settings may have better overall performance than the others, but none of them could dominate in terms of performance. In some settings, the restricted LP is more likely to become infeasible later as we round and fix more and more variables.

For the 20 real-size problem instances in this testing pool, we used the following setting: select one fractional  $z$  variable from each train to form a candidate pool. The candidate  $z$  variable is at the first control point where a train's path becomes fractional and has the largest time index among all fractional  $z$  variables at that control point. Then we pick one variable from the candidates with the smallest time index. If the solution has no fractional  $z$  variables, we pick the first fractional  $x$  variable instead. As for rounding direction, we use the hybrid method with  $[l_d, l_u] = [0.4, 0.6]$ . The computational results are presented in Table 4.7. The algorithm performed 24 to 73 iterations within 2 hours. In most iterations, the value of the selected variable is larger than 0.4, meaning that we either round it to 1 directly or evaluate the two options of rounding to 0 and rounding to 1. Although we only round one variable each time, the computational time varied greatly from 0.01 minute to 52 minutes. The algorithm terminated due to time limit for most of the instances. Four instances ended up with an infeasible restricted model. We only obtained a feasible solution for 5 instances, 4 of

which are easy instances. But even for these 5 instances, their final objective values were worse than those from the base model. Further, 3 out of these 5 instances also took longer time to solve than the base mode. Based on this testing result, we conclude that the rounding heuristic does not perform well for MP problem because of the long computational time per iteration and the lack of coordination between the selections of variables; the latter issue may lead to infeasibility in the restricted problem.

Instance	Base model		Base + constraint (4.13)				Base + constraint (4.14)				Base + constraint (4.13) and (4.14)			
	Cplex time (min)	# int path	# forcing constr	% obj incr	Cplex time (min)	# int path	# forcing constr	% obj incr	Cplex time (min)	# int path	# forcing constr	% obj incr	Cplex time (min)	# int path
E01	0.4	6	15,463	9%	0.4	6	15,007	8%	0.4	6	30,470	13%	0.5	15
E02	0.9	0	16,244	18%	0.6	2	15,741	34%	0.6	2	31,985	35%	0.7	18
E03	0.1	3	16,175	59%	0.5	3	15,810	41%	0.5	3	31,985	70%	0.5	9
E04	0.1	2	17,065	23%	0.2	2	16,597	23%	0.2	2	33,662	29%	0.3	2
E05	0.5	2	15,404	16%	0.2	2	15,179	23%	0.6	2	30,583	32%	0.6	2
E06	0.2	2	15,221	27%	0.1	2	14,771	29%	0.1	2	29,992	52%	0.1	2
E07	0.1	6	14,930	51%	0.1	6	14,599	46%	0.1	6	29,529	88%	0.2	6
E08	0.3	2	16,904	36%	0.6	2	16,513	5%	0.2	2	33,417	42%	0.7	13
E09	1.6	3	15,636	2%	0.7	3	15,058	2%	0.7	3	30,694	3%	0.8	3
E10	1.3	4	15,922	19%	1.1	4	15,526	15%	2.7	4	31,448	24%	2.6	4
H01	5	5	16,523	16%	6.6	5	16,013	11%	10.7	5	32,536	23%	22.4	5
H02	0.7	6	16,708	2%	0.6	17	16,387	2%	1	18	33,095	2%	0.7	17
H03	0.3	2	15,736	11%	0.3	2	15,266	6%	0.3	2	31,002	13%	0.3	2
H04	0.7	0	17,146	15%	1	0	16,754	28%	1.5	0	33,900	32%	2	0
H05	1.6	1	17,318	25%	4.3	1	16,902	25%	7.6	1	34,220	38%	11	1
H06	0.3	2	15,208	38%	0.6	2	14,722	51%	0.6	6	29,930	66%	0.7	6
H07	1.1	0	15,312	18%	1.7	0	15,011	12%	1.8	0	30,323	27%	3.5	0
H08	3.5	6	12,609	21%	8.5	6	12,052	10%	6	6	24,661	25%	12.8	6
H09	6.5	1	16,883	20%	7.8	1	16,350	6%	8.5	1	33,233	20%	11.9	1
H10	0.6	2	16,997	24%	0.9	2	16,550	46%	7.9	2	33,547	51%	5.8	2

Table 4.6 LP solution with and without headway-based forcing constraints

Instance	# iter	Rounding direction			Cplex time each iter (min)			Final obj	Total time (min)
		Round down	Round up	Evaluate	Min	Max	Avg		
E01	33	2	8	22	0.02	13.52	3.08	348	103
E02	39	2	20	16	0.02	8.70	1.17	376	47
E03	60	7	30	22	0.02	2.20	0.33	INF	21
E04	33	0	5	27	0.03	19.96	4.00	NFS	133
E05	41	0	1	39	0.02	42.18	4.23	NFS	175
E06	36	2	5	28	0.01	18.81	3.52	NFS	128
E07	48	2	14	31	0.02	8.85	0.80	685	41
E08	25	1	3	20	0.02	0.65	0.18	570	5
E09	33	1	9	22	0.02	38.97	3.50	NFS	121
E10	73	3	53	16	0.01	8.38	1.64	NFS	120
H01	39	3	15	20	0.03	18.28	3.45	NFS	135
H02	38	3	15	19	0.02	1.01	0.17	687	7
H03	35	0	6	28	0.02	20.21	2.59	INF	91
H04	37	5	18	13	0.02	6.60	1.04	INF	39
H05	31	3	19	8	0.03	22.92	4.30	NFS	134
H06	67	7	30	29	0.02	16.24	1.11	INF	75
H07	25	0	11	13	0.02	15.15	4.89	NFS	123
H08	24	1	15	7	0.03	23.38	5.89	NFS	142
H09	28	2	19	6	0.03	47.52	6.51	NFS	183
H10	26	1	4	20	0.02	51.87	6.03	NFS	158

Table 4.7 Performance of rounding heuristic

### 4.7.3 Directional movement planning

In directional movement planning, we first solve a subproblem with trains in the selected direction only. For this test, we chose the direction with more passing conflicts; ties are broken by choosing the direction with more trains. We imposed a time limit of 1 hour for the first subproblem and remaining time till 2 hours for later iterations. The re-routing procedure was applied at the end of each iteration to refine the (partial) solution. Also, we used the solution from a previous iteration to warm start

CPLEX solver in the current iteration. Table 4.8 presents the computational result in each iteration and the overall performance. The algorithm terminated within 4 iterations for most instances, solving for each direction twice. Half of the instances started with direction 0 (eastbound) and the other half started with direction 1 (westbound). As we start with the direction with more conflicts and more trains, the first iteration usually took longer time to solve. In particular, 4 instances (E09, H06, H08, and H09) did not finish their first iteration within 1 hour. On the contrast, the second and fourth iterations were very easy to solve (solution time was less than 2 minutes). Overall, this heuristic performed very well. For the 10 easy instances, the heuristic obtained optimal solutions for 4 instances and near-optimal solutions for another 4 instance, all with shorter time. It also obtained better solutions for all hard instances, although the computational time reached the 2-hour limit for 2 of them.

#### **4.7.4 Expanding planning horizon**

For the horizon expansion algorithm, we use the following parameters: (1) set the initial horizon or extend the horizon by counting 10 trains or 5 passing conflicts, whichever is earlier; (2) if a subproblem is solved, fix 50% of the new movements; (3) if a subproblem is not solve, reduce the horizon to be 80% of the current horizon. The new horizon ( $L_{sub} - L_{fix}$ ) should be at least 1.2 hours long and  $L_{sub}$  should be at least half an hour longer than  $L_{solved}$ . We use 1.2 hours as the minimum horizon length because trains take 2.4 hours, on average, to traverse the entire territory; (4) set a time limit of 10 minutes on each subproblem and 2 hours for the entire algorithm.

The results are presented in Table 4.9. Each instance took 2 to 9 iterations to be

solved. An iteration typically solved a horizon length,  $L_{sub} - L_{fix}$ , of 2 to 3 hours; the average CPLEX solution time per iteration varies in the range of 0.2 to 6.8 minutes. Compared to the base model, this heuristic solved all problem instances within a shorter time period except for E01. In particular, the method obtained good solutions within 1 hour with improved objective values for all hard instances except for H02 and H03. Further, all solutions were optimal or near optimal for the easy instances.

Instance	Start dir	# iter	Result each iteration										Final obj	Total time (min)
			Iter 1		Iter 2		Iter 3		Iter 4		Iter 5			
			Soln gap	Cplex time (min)	Soln gap	Cplex time (min)	Soln gap	Cplex time (min)	Soln gap	Cplex time (min)	Soln gap	Cplex time (min)		
E01	0	3	1%	1.2	1%	0.6	1%	1.3					366	3
E02	1	4	1%	0.8	0%	0.3	0%	0.5	0%	0.2			345	2
E03	0	4	1%	8.7	0%	0.3	1%	1.3	0%	0.2			585	11
E04	1	3	1%	1.2	0%	0.6	1%	1.5					314	4
E05	0	4	0%	0.3	1%	0.9	0%	0.5	0%	0.5			365	3
E06	0	4	0%	0.5	0%	0.8	0%	0.4	0%	0.3			495	2
E07	0	4	1%	13.7	0%	0.5	1%	4	0%	0.2			568	19
E08	1	5	0%	0.7	0%	0.9	0%	0.7	1%	0.5	0%	0.7	570	4
E09	1	4	6%	60	0%	0.5	1%	1.7	0%	0.3			451	63
E10	0	4	0%	0.2	1%	1.5	0%	0.4	1%	1			594	4
H01	1	4	1%	12	0%	0.9	1%	1.6	0%	0.3			1,117	16
H02	0	4	0%	0.9	1%	1.7	1%	0.9	1%	0.9			571	5
H03	0	4	1%	2.6	1%	1.3	1%	1	1%	0.7			481	6
H04	1	4	1%	1.7	1%	1.6	1%	1.4	1%	1			763	6
H05	1	4	1%	8	1%	1.2	1%	2.9	1%	0.7			955	14
H06	1	4	44%	71.9	0%	0.6	1%	5	0%	0.4			743	81
H07	0	4	1%	32.8	0%	0.7	1%	7.3	0%	0.4			1,005	42
H08	1	3	44%	69	0%	0.5	8%	49.5					1,075	125
H09	0	3	28%	60.1	1%	0.7	26%	58.9					1,149	120
H10	1	3	0%	1	1%	1.3	1%	1.1					707	4

Table 4.8 Result of directional movement planning

#### 4.7.5 Sequential conflict resolution heuristic

The sequence conflict resolution heuristic, as discussed in Section 4.6.1, has three variations to solve conflicts of train movements. We tested all three methods. In the lookahead method, we look out 2 hours to compare different passing and yielding decisions for each conflict. The randomization method was repeated 50 times to generate multiple solutions and pick the best one. For each initial solution generated by the three methods, we applied the re-routing procedure for local improvement. We report the final objective value and solution time of each method in Table 4.10. The solution time included the time for both initial solution construction and re-routing, but did not include the time to prepare instance and export solution. As we can see, the rule-based method only took less than 1 second. The lookahead method usually needed more time, but the solution time was still within 10 seconds. The total computational time for all methods, plus other related operations, was within 30 seconds for all instances except for H08. Note that the total time tended to be slightly higher for the hard instances, but the difference was not as obvious as the optimization-based heuristics. The last column of Table 4.10 reports the best solution from the three methods. As we expected, the lookahead method and randomization method (when repeated multiple times) performed better than the rule-based method. In practice, we can always run all methods in parallel and select the best solution.

Instance	# iter	Avg hrz length per iter (hr)	Avg Cplex time per iter (min)	Final obj	Total time (min)
<b>E01</b>	6	2.8	4.8	307	26
<b>E02</b>	4	2.6	0.6	366	3
<b>E03</b>	5	2.2	0.2	566	3
<b>E04</b>	3	3.0	4.0	368	10
<b>E05</b>	3	3.2	0.7	365	2
<b>E06</b>	4	2.7	1.1	490	4
<b>E07</b>	2	4.0	4.0	555	10
<b>E08</b>	3	3.0	1.7	570	6
<b>E09</b>	3	3.1	2.1	442	6
<b>E10</b>	5	2.1	0.6	560	3
<b>H01</b>	7	1.6	0.2	1,164	2
<b>H02</b>	6	3.2	6.8	656	37
<b>H03</b>	8	2.6	6.7	597	51
<b>H04</b>	9	2.4	5.3	793	53
<b>H05</b>	7	1.7	3.1	1,034	20
<b>H06</b>	5	2.0	2.1	749	11
<b>H07</b>	5	2.4	2.0	986	17
<b>H08</b>	6	1.7	1.1	1,129	6
<b>H09</b>	9	1.7	4.6	1,191	38
<b>H10</b>	5	2.1	0.8	707	4

Table 4.9 Result summary for expanding horizon heuristic

Finally, we summarize the computational performance, in terms of objective value and solution time, of the base model and all heuristic methods in Table 4.11. To see the overall performance of the methods, we define a solution whose objective value is within 10% gap of the lowest objective value among all methods and solution time is within 30 minutes as a “good” solution. Note that the lowest objective value was optimal for the easy instances but not necessarily for the hard ones. We marked all solutions meeting this definition with dark shades in Table 4.11. Overall, the directional

movement planning method and expanding horizon methods performed best among all heuristics. The LP-based time window method and decomposition by train group method had quite good performance for the easy instances but not for the hard ones. The standalone heuristic ran very fast, but its solution quality was not the best. The rounding heuristic failed to find feasible solutions for many instances; its performance was not even as good as the base model.

Instance	Rule-based		Lookahead		Randomized		Best obj	Total time (sec)	Best method
	Obj	Time (sec)	Obj	Time (sec)	Best rand obj	Time (sec)			
<b>E01</b>	646	0.1	472	1.2	372	7.2	372	10	Randomized
<b>E02</b>	790	0.2	498	1.4	354	12.5	354	16	Randomized
<b>E03</b>	1,569	0.3	929	4.5	940	14.3	929	21	Lookahead
<b>E04</b>	445	0.2	445	1.5	372	12.8	372	17	Randomized
<b>E05</b>	939	0.2	852	7.0	800	12.1	800	21	Randomized
<b>E06</b>	1,584	0.2	891	2.6	1,064	11.1	891	16	Lookahead
<b>E07</b>	640	0.2	640	3.0	609	12.6	609	17	Randomized
<b>E08</b>	808	0.2	677	3.3	572	12.4	572	18	Randomized
<b>E09</b>	778	0.4	694	3.0	630	17.1	630	22	Randomized
<b>E10</b>	743	0.2	607	3.6	603	12.6	603	18	Randomized
<b>H01</b>	1,908	0.4	1,466	6.7	1,484	19.3	1,466	29	Lookahead
<b>H02</b>	1,203	0.3	685	2.1	713	17.0	685	21	Lookahead
<b>H03</b>	1,034	0.4	604	3.6	587	16.4	587	22	Randomized
<b>H04</b>	928	0.2	868	3.5	884	17.0	868	23	Lookahead
<b>H05</b>	1,408	0.4	1,173	5.0	1,201	19.9	1,173	27	Lookahead
<b>H06</b>	1,202	0.9	918	5.0	845	14.6	845	26	Randomized
<b>H07</b>	1,524	0.3	1,258	4.1	1,359	12.3	1,258	19	Lookahead
<b>H08</b>	INF	0.2	1,372	6.4	1,558	28.2	1,372	37	Lookahead
<b>H09</b>	1,536	0.4	1,265	4.9	1,220	16.4	1,220	24	Randomized
<b>H10</b>	1,114	0.3	1,037	7.5	905	19.2	905	29	Randomized

Table 4.10 Solution of different conflict resolution strategies



Instance	Base		Time window mtd		Train grp decomp		Rounding		Directional MP		Expanding horizon		Conflict resolution	
	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)	Obj	Time (min)
E01	307	17	308	3	307	13	348	103	366	3	307	26	372	0.2
E02	345	26	345	5	345	22	376	47	345	2	366	3	354	0.3
E03	508	27	508	3	686	34	INF	21	585	11	566	3	929	0.4
E04	314	32	314	3	314	11	NFS	133	314	4	368	10	372	0.3
E05	365	32	365	9	365	26	NFS	175	365	3	365	2	800	0.4
E06	479	52	712	21	485	63	NFS	128	495	2	490	4	891	0.3
E07	545	55	560	4	561	21	685	41	568	19	555	10	609	0.3
E08	570	72	600	2	570	25	570	5	570	4	570	6	572	0.3
E09	441	94	453	11	441	30	NFS	121	451	63	442	6	630	0.4
E10	560	107	560	12	577	120	NFS	120	594	4	560	3	603	0.3
H01	1,396	122	1,109	50	NFS	123	NFS	135	1,117	16	1,164	2	1,466	0.5
H02	634	121	560	50	571	24	687	7	571	5	656	37	685	0.4
H03	558	129	481	45	NFS	121	INF	91	481	6	597	51	587	0.4
H04	989	120	741	120	NFS	121	INF	39	763	6	793	53	868	0.4
H05	1,485	121	8,344	122	NFS	122	NFS	134	955	14	1,034	20	1,173	0.5
H06	826	121	732	2	738	120	INF	75	743	81	749	11	845	0.4
H07	3,247	120	996	120	NFS	120	NFS	123	1,005	42	986	17	1,258	0.3
H08	7,676	122	5,271	123	NFS	120	NFS	142	1,075	125	1,129	6	1,372	0.6
H09	10,275	121	5,891	121	NFS	120	NFS	183	1,149	120	1,191	38	1,220	0.4
H10	12,393	120	793	120	NFS	120	NFS	158	707	4	707	4	905	0.5

Table 4.11 Solution summary for all methods

## 4.8 CONCLUSION

Train movement planning is an important task for railroad companies, especially for those whose train schedules are ad hoc and a train can move on all parallel tracks on its route. An effective plan of meeting and passing events between trains with different direction, speed, and priority can significantly improve the fluidity of traffic flow and increase the transportation volume. In this work, we study the movement planning

problem in a general train dispatching territory with one or more through tracks and multiple sidings. We propose a discrete-time IP model that can coordinate the movements of trains within a given planning horizon such that the overall weighted train delay is minimized. To solve this problem, we proposed a number of optimization-based heuristics and a standalone conflict resolution algorithm. In particular, the standalone heuristic can provide a good feasible solution within seconds. The directional movement planning method, which is best used for multi-track territories, explores the benefit of maintaining directional traffic and only utilizes crossovers when it helps to significantly reduce train delay. The expanding horizon method, on the other hand, gradually extends the planning horizon and partially fixes the solution. These two approaches can provide near-optimal solutions within reasonable amount of time.

## Appendix A: Forcing Constraints of Indicators

In Section 2.4.2.2, we introduced three sets of indicator variables,  $u_{i_1 i_2}^1, u_{i_1 i_2}^2$ , and  $u_{i_1 i_2 j}^3$  to linearize the Push-Pull conditions (2.11a) and (2.11b). The following forcing constraints (A.1) to (A.5) establish the relationships between the indicator variables and the crew assignments decisions.

$$u_{i_1 i_2}^1 = \begin{cases} 0, & \text{if } \sum_{i_1 \prec i \prec i_2} v_i^H < \beta_s \\ 1, & \text{if } \sum_{i_1 \prec i \prec i_2} v_i^H \geq \beta_s \end{cases} \Rightarrow \begin{cases} \sum_{i_1 \prec i \prec i_2} v_i^H \geq \beta_s u_{i_1 i_2}^1, & \forall i_1 \prec i_2 \\ \sum_{i_1 \prec i \prec i_2} v_i^H \leq (\beta_s - 1) + M u_{i_1 i_2}^1, & \forall i_1 \prec i_2 \end{cases} \quad (\text{A.1})$$

$$u_{i_1 i_2}^2 = \begin{cases} 0, & \text{if } v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H \leq \beta_s \\ 1, & \text{if } v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H > \beta_s \end{cases} \Rightarrow \begin{cases} v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H \geq (\beta_s + 1) u_{i_1 i_2}^2, & \forall i_1 \prec i_2 \\ v_{i_1}^H + \sum_{i_1 \prec i \prec i_2} v_i^H \leq \beta_s + M u_{i_1 i_2}^2, & \forall i_1 \prec i_2 \end{cases} \quad (\text{A.3})$$

$$u_{i_1 i_2 j}^3 = 1, \text{ if } \sum_{j' \in CR(i_1, i_2), td_{j'} < td_j} x_{i_1 j'}^H \geq 1 \Rightarrow \sum_{j' \in CR(i_1, i_2), td_{j'} < td_j} x_{i_1 j'}^H \leq M u_{i_1 i_2 j}^3, \forall i_1 \prec i_2, j \in CR(i_1, i_2) \quad (\text{A.5})$$

## References

- Abbink, E., M. Fischetti, L. Kroon, G. Timmer, M. Vromans. 2005. Reinventing crew scheduling at Netherlands railways. *Interfaces* **35**(5) 393–401.
- Association of American Railroads (AAR). 2015(a). Freight railroad capacity and investment. <https://www.aar.org/BackgroundPapers/Freight%20Railroad%20Capacity%20and%20InvestmeIn.pdf>. Accessed March 2015.
- Association of American Railroads (AAR). 2015(b). Freight rail traffic data. <https://www.aar.org/data-center/rail-traffic-data>. Accessed May 2015.
- Balakrishnan, A., T.L. Magnanti, R.T. Wong. 1989. A dual-ascent procedure for large-scale uncapacitated network design. *Oper. Res.* **37**(5) 716–740.
- Balakrishnan, A., S. Lin, A. Uygur. 2012. Optimization-based decision support system for train dispatching. Working paper.
- Barlow, R.E., K.D. Heidtmann. 1984. Computing  $k$ -out-of- $n$  system reliability. *Reliability, IEEE Transactions on* **33**(4) 322–323.
- Barnhart, C., A.M. Cohn, E.L. Johnson, D. Klabjan, G.L. Nemhauser, P.H. Vance. 2003. Airline crew scheduling. R.W. Hall, eds. *Handbook of Transportation Science*, 2<sup>nd</sup> edition. Kluwer Academic, Norwell, 517–560.
- Brännlund, U., P.O. Lindberg, A. Nöu, J.E. Nilsson. 1998. Railway timetabling using Lagrangian relaxation. *Trans. Sci.* **32**(4) 358–369.
- Cacchiani, V., A. Caprara, P. Toth. 2008. A column generation approach to train timetabling on a corridor. *4OR* **6**(2) 125–142.
- Cacchiani, V., A. Caprara, P. Toth. 2010. Scheduling extra freight trains on railway networks. *Trans. Res. Part B* **44** 215–231.
- Caprara, A., M. Fischetti, P. Toth, D. Vigo, P.L. Guida. 1997. Algorithms for railway crew management. *Math. Programming* **79**(1-3) 125–141.
- Caprara, A., P. Toth, D. Vigo, M. Fischetti. 1998. Modeling and solving the crew rostering problem. *Oper. Res.* **46**(6) 820–830.
- Caprara, A., M. Monaci, P. Toth. 2001. A global method for crew planning in railway applications. S. Voß, J.R. Daduna, eds. *Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems*, Vol. 505. Berlin: Springer, 17–36.
- Caprara, A., M. Fischetti, P. Toth. 2002. Modeling and solving the train timetabling problem. *Oper. Res.* **50** 851–861.
- Caprara, A., M. Monaci, P. Toth, P.L. Guida. 2006. A Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Appl. Math.* **154** 738–753.

- Caprara, A., L. Kroon, M. Monaci, M. Peeters, P. Toth. 2007. Passenger railway optimization. C. Barnhart, G. Laporte, eds. *Transportation, Handbooks in Operations Research and Management Science*, Vol. 14. Elsevier, Amsterdam, 129–187.
- Carey, M.. 1994a. A model and strategy for train pathing with choice of lines, platforms, and routes. *Trans. Res. Part B* **28**(5) 333–353.
- Carey, M.. 1994b. Extending a train pathing model from one-way to two-way track. *Trans. Res. Part B* **28**(5) 395–400.
- Carey, M., D. Lockwood. 1995. A model, algorithms and strategy for train pathing. *J. Oper. Res. Society* **46**(8) 988–1005.
- Chen, A.I., S.C. Graves. 2013. Inventory strategies for international non-profit healthcare organizations. *INFORMS 2013 Annual Meeting, Minneapolis*.
- Chung, J.W., S.M. Oh, I.C. Choi. 2009. A hybrid genetic algorithm for train sequencing in the Korean railway. *Omega* **37** 555–565.
- Clausen, J., A. Larsen, J. Larsen, N.J. Rezanova. 2010. Disruption management in the airline industry— concepts, models and methods. *Computers & Operations Research* **37** 809–821.
- Cordeau, J., P. Toth, D. Vigo. 1998. A survey of optimization models for train routing and scheduling. *Trans. Sci.* **32** (4) 380–404.
- Corman, F., A. D’Ariano, D. Pacciarelli, M. Pranzo. 2010. A tabu search algorithm for rerouting trains during rail operations. *Trans. Res. Part B* **44**(1) 175–192.
- D’Ariano, A., D. Pacciarelli, M. Pranzo, 2007. A branch and bound algorithm for scheduling trains in a railway network. *Euro. J. Oper. Res.* **183**(2) 643–657.
- D’Ariano, A., D. Pacciarelli, M. Pranzo. 2008a. Assessment of flexible timetables in real-time traffic management of a railway bottleneck. *Trans. Res. Part C* **16** 232–245.
- D’Ariano, A., F. Corman, D. Pacciarelli, M. Pranzo. 2008b. Reordering and local rerouting strategies to manage train traffic in real-time. *Trans. Sci.* **42** (4) 405–419.
- Dessouky, M., Q. Lu, J. Zhao, R.C. Leachman. 2006. An exact solution procedure to determine the optimal dispatching times for complex rail networks. *IIE transactions* **38**(2) 141–152.
- Ehrgott, M., D.M. Ryan. 2002. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis* **11**(3) 139–150.
- Ernst, A.T., H. Jiang, M. Krishnamoorthy, H. Nott, D. Sier. 2001. An integrated optimization model for train crew management, *Ann. Oper. Res.* **108**(1-4) 211–224.
- Federal Railroad Administration (FRA). 2012. National Rail Plan Progress Report. 14.

- Freling, R., R.M. Lentink, A.P.M. Wagelmans. 2004. A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm, *Ann. Oper. Res.* **127** 203–222.
- Gopalakrishnan, B., E.L. Johnson. 2005. Airline crew scheduling: State-of-the-art, *Ann. Oper. Res.* **140**(1) 305–337.
- Gorman, M.F., M. Sarrafzadeh. 2000. An application of dynamic programming to crew balancing at Burlington Northern Santa Fe Railway, *Int. J. Services Technology and Management* **1** 174–187.
- Harrod, S. 2011. Modeling network transition constraints with hypergraphs. *Trans. Sci.* **45**(1) 81–97.
- Hong, Y. 2013. On computing the distribution function for the Poisson binomial distribution. *Computational Statistics & Data Analysis.* **59** 41–51.
- IBM ILOG CPLEX Optimization Studio V12.3 documentation, IBM Corporation. 2011. User's Manual for CPLEX. <http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r3/index.jsp>. Accessed July 2012.
- Ionescu, L., N. Kliewer. 2011. Increasing flexibility of airline crew schedules. *Procedia Social and Behavioral Sciences* **20** 1019–1028.
- Jütte, S., M. Albers, U.W. Thonemann, K. Haase. 2011. Optimizing railway crew scheduling at DB Schenker, *Interfaces* **41**(2) 109–122.
- Krüger, N.A., I. Vierth, F. Fakhraei Roudsari. 2013. Spatial, temporal and size distribution of freight train delays: evidence from Sweden. Working paper, CTS. <http://www.transportportal.se/swopec/CTS2013-8.pdf>. Accessed November 2014.
- Liu, S.Q., E. Kozan. 2009. Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers & Oper. Res.* **36** 2840–2852.
- Liu, S.Q., E. Kozan. 2011. Scheduling trains with priorities: a no-wait blocking parallel-machine job-shop scheduling model. *Trans. Sci.* **45**(2) 175–198.
- Lu, Q., M. Dessouky, R.C. Leachman. 2004. Modeling train movements through complex rail networks. *ACM Transactions on Modeling and Computer Simulation* **14** (1), 48–75.
- Lusby, R.M., J. Larsen. M. Ehrgott, D. Ryan. 2011. Railway track allocation: models and methods. *OR Spectrum* **33**(4) 843–883.
- Mascis, A., D. Pacciarelli. 2002. Job-shop scheduling with blocking and no-wait constraints. *Euro. J. Oper. Res.* **143** 498–517.
- Mu, S., M. Dessouky. 2011. Scheduling freight trains traveling on complex networks. *Trans. Res. Part B* **45** 1103–1123.
- Nemani, A.K., R.K. Ahuja. 2010. OR models in freight railroad industry. J.J. Cochran, eds. *Wiley Encyclopedia of Operations Research and Management Science*, John

- Wiley & Sons, Inc..
- Rosenberger, J.M., A.J. Schaefer, D. Goldsman, E.L. Johnson, A.J. Kleywegt, G.L. Nemhauser. 2002. A stochastic model of airline operations. *Transportation Science* **36**(4), 357–377.
- Şahin, G., R.K. Ahuja, C.B. Cunha. 2008. Integer programming based approaches for the train dispatching problem. [http://research.sabanciuniv.edu/19388/1/TrainDispatching\\_SahinAhujaCunha%282008-02-23%29.pdf](http://research.sabanciuniv.edu/19388/1/TrainDispatching_SahinAhujaCunha%282008-02-23%29.pdf). Accessed July 2012.
- Şahin, G., B. Yüceoğlu. 2011. Tactical crew planning in railways, *Transportation Res. Part E: Logistics* **47**(6) 1221–1243.
- Schaefer, A.J., E.L. Johnson, A.J. Kleywegt, G.L. Nemhauser. 2005. Airline crew scheduling under uncertainty. *Transportation Science* **39**(3), 340–348.
- Shebalov, S., D. Klabjan. 2006. Robust airline crew pairing: Move-up crews. *Transportation Science* **40**(3) 300–312.
- Surface Transportation Board (STB), U. S. Department of Transportation. 2013. Annual Report Financial Data. <http://www.stb.dot.gov/econdata.nsf/f039526076cc0f8e8525660b006870c9?OpeView>. Accessed March 2015.
- Tam, B., M. Ehrgott, D. Ryan, G. Zakeri. 2011. A comparison of stochastic programming and bi-objective optimization approaches to robust airline crew scheduling. *OR Spectrum* **33**(1), 49–75.
- Vaidyanathan, B., K.C. Jha, R.K. Ahuja. 2007. Multicommodity network flow approach to the railroad crew-scheduling problem, *IBM J. Res. Dev.* **51**(3) 325–344.
- Veelenturf, L.P., D. Potthoff, D. Huisman, L.G. Kroon. 2012. Railway crew rescheduling with retiming. *Transportation Research Part C* **20**(1) 96–110.
- Yen, J.W., J.R. Birge. 2006. A stochastic programming approach to the airline crew scheduling problem. *Transportation Science* **40**(1), 3–14.